Исследование ускоренного поиска близких текстовых последовательностей с помощью векторных представлений

Артем Соколов
Международный научно-учебный центр информационных технологий и систем

1 Введение

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

Для сравнения последовательностей необходимо задаться метрикой, которая адекватна задаче и эффективно вычислима. Первому условию часто удовлетворяет расстояние Левенштейна (классическое расстояние редактирования) [3]. Оно определяется между символьными строками x, 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, а также хорошо описывают 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]. Из-за 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].

Нами разработан [6] метод оценки расстояния Левенштейна на основе вложения расстояния редактирования в векторное пространство. Также в [6] был 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 данной работе описаны результаты численных экспериментов по проверке теоретических результатов предложенной схемы на искусственных данных (раздел 3). Также рассматривается применение предложенного алгоритма приближенного нахождения ближайших строк к следующим актуальным практическим задачам (раздел 4) фильтрации дубликатов (в поисковых машинах, системах документооборота и др.) и обнаружения спама (для веб-страниц, электронной почты и т.п.).

2 Вложение расстояния редактирования

В данном разделе 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 $x \in \Sigma^n$ и $q \in N$ назовем q-граммой подстроку длиной q. Назовем q-граммным вектором вектор вида $v_{n,q}(x) \in (N \cup \{0\})^{|\Sigma|}$, где каждой q-грамме $\sigma \in \Sigma^q$ соответствует элемент вектора $(v_{n,q}(x))_\sigma \in N \cup \{0\}$, значение которого – число встречаемости 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 задано 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

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

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

В качестве расстояния между такими векторами примем q-граммное расстояние, и на его основании определим новое расстояние между строками $x,y\in\Sigma^w$:

$$d^{\Sigma}_{w,q1,q2}(x,y) = \sum_{q=q1,\ldots,q2} d_q(x,y). \quad (2)$$

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

$$D(x,y) = \sum_{i=1,\ldots,n-w+1} d^{\Sigma}_{w,q1,q2}(x[i,i+w-1],y[i,i+w-1])\} / ((n-w+1)(\Delta q+1)). \quad (3)$$

Пусть $q_1 = 2w/3$, $\Delta q = \lfloor ½(-7+(57+16(w-q_1))^{½})\rfloor$, $Q=(\Delta q+1)(\Delta q+2)$, $t=w-\Delta q+1$. В [6] показано, что

Если $ed(x,y) > k_2$, то $D(x,y) \geq Qt(k_2 / (2(\Delta q+1))-2) / ((n-w+1)(\Delta q+1)) = d_2$. (4)
Если $ed(x,y) \leq k_1$, то $D(x,y) \leq 2k_1[w^2+(n+1)] / (n-w+1) = d_1$. (5)

При выполнении условия $d_2>d_1$, чем больше разность между k_1 и k_2, тем точнее можно аппроксимировать $ed(x,y)$ при известном $D(x,y)$. Положим $w=n^{\gamma}$, тогда обеспечивающий наибольшую точность аппроксимации показатель роста w достигается при $\gamma=0.5$. При этом $k_2 = \Omega(k_1 n^{1/2})$, время построения векторов $v(\cdot)$ – $O(n^{3/2})$, 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, менее требовательная к ресурсам.

2.2 Рандомизированная схема

Пусть вектор, обозначаемый $v^i_{w,q1,q2}(x)$, является конкатенаций q-граммных (от q_1 до q_2) векторов (2) подстроки $x[i,i+w-1]$ длиной w, где i выбрано случайно и равновероятно из множества возможных позиций окна шириной w: $i = 1,...,n-w+1$. Размерность вектора $v^i_{w,q1,q2}(x)$ равна $\sum_{q=q1,\ldots,q2} \Sigma^q$.

Пусть φ^i – случайный вектор такой же размерности, как и $v^i_{w,q1,q2}(x)$, с элементами, выбранными случайно из распределения Коши $p(x)=(\pi(1 + x^2))^{-1}$. Построим для строки x хеш-вектор размерностью K: $h(x) = (h_1(x),h_2(x),...,h_K(x))$, где

$$h_i(x) = \lfloor ((v^i_{w,q1,q2}(x), \varphi_i) + b_i) / r \rfloor, \quad (6)$$

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

Если определить шар $B(q, k) = \{x \in \Sigma^n \mid ed(q,x) \leq k\}$, то семейство $H=\{h: \Sigma^n \rightarrow X\}$ (где X – некоторое конечное или счетное множество значений) называется [7] (k_1,k_2,p_I,p_{II})-чувствительным или просто локально-чувствительным, если для любых $x,y\in \Sigma^n$ и любой независимо и равновероятно выбранной хеш-функции $h \in H$ выполняется:

если $x \in B(y,k_1)$, то $Prob[h(x) = h(y)] > p_I$, (7)
если $x \notin B(y,k_2)$, то $Prob[h(x) = h(y)] < p_{II}$. (8)

Для того, чтобы семейство H позволяло отличить «близкие» строки от «далеких», необходимо, чтобы выполнялось условие $k_1 < k_2$, т.е. «близкая» строка x должна находится ближе к y, чем строка,

считающаяся «дальней», и $p_{II} < p_I$, 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, $t=w-\Delta q+1$ 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 штук K-мерных случайных хеш-векторов $h^j(x) = (h^j_1(x), h^j_2(x),..., h^j_K(x))$, $j = 1,...,L$.
2. Для каждого уникального хеш-вектора, полученного из всех строк базы $h^j(x)$, $j = 1,...,L$, $x \in P$ создается ячейка памяти. Общее число ячеек $C <= L|P|$.
3. Каждая строка базы P ставится в соответствие тем ячейкам, хеш-векторы которых ею продуцируются.

Когда необходимо найти приближенный ближайший сосед к поступившей строке $q \in \Sigma^n$, то:
1. Формируется L ее хеш-векторов.
2. Просматриваются ячейки, хеш-векторы которых полностью совпадают со сформированными для q, соответствующие каждому хеш-вектору $h^j(q)=(h^j_1(q), h^j_2(q),..., h^j_K(q))$, $j = 1,...,L$.
3. Составляется список содержащихся в просмотренных ячейках строк из P, обозначаемый S. Так как одна и та же строка P может встречаться в S несколько раз, S можно представить в виде мультимножества.

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

В [6] определяются значения вероятностей p_I и p_{II} и доказывается следующая теорема:

> **Теорема 1.** При описанном построении векторов, значении $\rho = \log(p_I/p_{II})$ и при следующих значениях K, L:
>
> $$K = \log_{1/p_{II}} |P|, \quad (9)$$
> $$L = |P|^{\rho}. \quad (10)$$
>
> алгоритм поиска выдаст в S с вероятностью большей, чем 1/2, строку y, такую, что $ed(x,y) \le k_2$, где $k_2 = O(zn^{1/3}\ln n)$, z –параметр, которые влияет на значение k_2 и значения вероятностей p_I, p_{II}.

Для экспериментов, описываемых ниже, были зафиксированы 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 мы использовали модификацию описанной схемы LSH, называемую LSH-лес, которая позволяет находить ближайших соседей без обновления хеш-векторов при изменении размера базы P или параметра k_2 [11].

Для каждого $j=1,...,L$ 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 – 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 у хеш-векторов 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:

1. Формируются L его K-мерных хеш-векторов $h^j(q)$.
2. Для каждого хеш-вектора $h^j(q)$ в T_j находится узел, соответствующий $h^j(p)$ с наибольшим числом совпадающих первых элементов этих векторов.
3. Начиная от найденных в 2 узлов, все L деревьев синхронно проходятся по направлению к корню, и соответствующие указанным узлам строки p добавляется в результирующее мультимножество строк S.
4. 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], аналогично теореме 1, S будет содержать приближенных ближайших соседей к запросу с ненулевой константной вероятностью.

3 Численное исследование теоретических ограничений

3.1 Экспериментальный материал (база) RandomStrings

Для проверки теоретических результатов мы провели 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 намного меньше длины строки, а посчитать расстояние редактирование не представляется возможным из-за вычислительной сложности, положим, что расстояние редактирования примерно равно сумме количеств искажений. Аналогичное допущение было принято в [12]. В дальнейшем мы будет 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) (3) в интервал (4) и (5), выполнялось для строк из базы RandomStrings. На рис. 1 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. Теоретические значения верхней (4) и нижней границ (5) на значения расстояния изображены, соответственно, сплошной и пунктирной тонкими линиями. Видно, что экспериментальные данные соответствуют теоретическим оценкам.

Выполаживание графика происходит 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 $q = q_1,\ldots,q_2$.

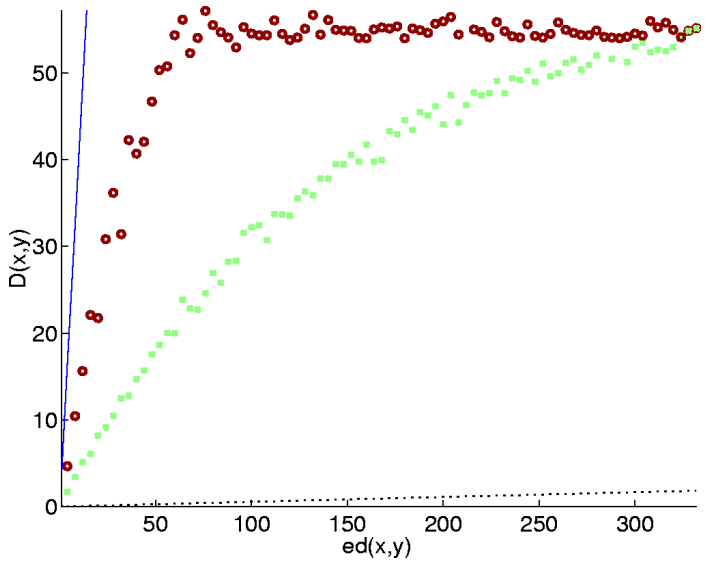


Рис. 1. Типичная зависимость расстояния D(x,y) от ed(x,y). Для данных рисунков n = 5000 и n = 10000.

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 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 теории.

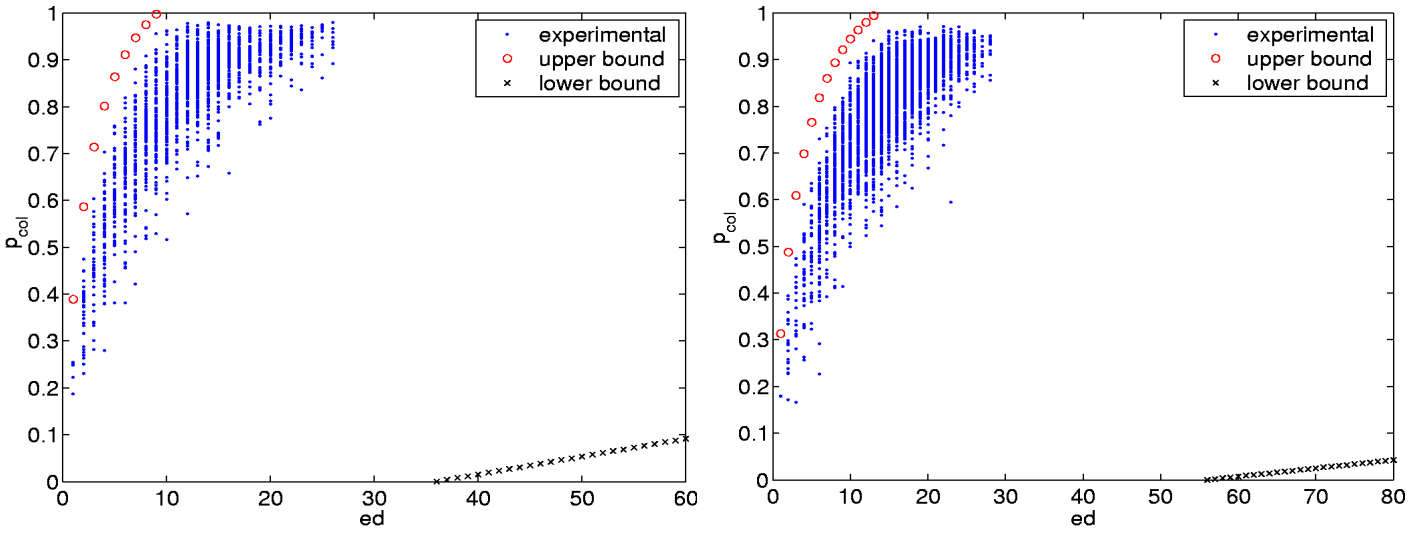


Рис. 2. Экспериментальные значения хеш-функций $h_i(x)$ (6) в зависимости от расстояния редактирования между строками длины 1000 и 3000 символов.

Значение L и дополнительная фильтрация

На практике, так как необходимые ресурсы растут линейно от L (9), рекомендуемые теорией значения L слишком велики [11].

Однако при использовании меньших значений теория не гарантирует, что в возвращенное мультимножество строк S попадет как минимум одна строка y, такая, что $ed(q,y) \le k_2$. 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 false positives (т.е. строки с $ed(q,y) > k_2$).

Проводились эксперименты со следующими способами оценки качества S:

- по значению точности (п. A);
- по упорядоченности S относительно известного эталона (п. B).

A. Эксперимент по оценке качества S по значению точности

Традиционно в системах поиска для исследования качества множества документов, возвращаемого на запрос, анализируется компромисс между количеством true positives (т.е. количество релевантных текстов) и false positives (т.е. количеством нерелевантных текстов). 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 точность от полноты [13].

Здесь (см. формулировку задачи в разделе 2.2) данный подход оказывается неинформативным, поскольку рассматриваемая задача поиска приближенного ближайшего соседа подразумевает нахождение одного соседа, а не множества соседей. Поэтому не важно отношение $|B(q, k_2) \cap P \cap S| / |B(q,k_2) \cap P|$ (полнота).

Ситуация с неинформативностью полноты возникает и в оценке качества поисковых систем: для пользователя все множество возвращенных результатов неинтересно, он обычно ограничивается просмотром только первых страниц результатов [14]. В таких случаях вместо точности и полноты обычно используют характеристику “точность на уровне n” (“precision at n / P@n”) [15], 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 $|B(q, k_2) \cap P \cap S| / |S|$.

Чтобы не вносить смещение в оценку точности на уровне |S| из-за неодинакового количества строк внутри и вне шара $B(q, k_2)$ и разного количества строк на определенном расстоянии от q, мы отобрали из набора RandomStrings (раздел 3) 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 на каждом расстоянии 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 процедуры п. 2.2. На вход процедуры поиска приближенно ближайшего соседа подавался запрос q. По полученным 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 1.

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

Табл. 1. Точность на уровне |S| в зависимости от значения L и |S|.

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 изменятся, 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 рекомендованом значении 2L.

B. Эксперимент по сравнению 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: $S'=\{ed(x,q) \mid x \in S\}$.

Для сравнения упорядоченных мультимножеств возьмем за основу обобщенное расстояние Кэндалла [16], 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 $M_1 \cup M_2$:

$$D_K(M_1, M_2) = \sum_{i,j \in M1 \cup M2} u(i,j) \qquad (11)$$

Мы изменили описанный алгоритм подсчета обобщенного расстояния Кэндалла 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 в хотя бы 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: $S'_1 \rightarrow M_1 = \{$”$1_1$”, ”$2_1$”, ”$3_1$”, ”$4_1$”, ”$4_2$”, ”$3_2$”, ”$5_1$”$\}$, $S'_2 \rightarrow M_2 = \{$”$1_1$”, ”$3_1$”, ”$3_2$”, ”$4_1$”, ”$4_2$”, ”$3_3$”, ”$4_3$”$\}$. 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.

Аналогично эксперименту по вычислению точности на уровне |S| (п. A), количество строк в S, возвращаемых процедурой LSH-лес и фигурирующее в описании алгоритма как 2L, было переменным.

То есть алгоритм возвращал указанное количество строк, полученных в результате этапа подъема по LSH-деревьям.

Исследование проведено на наборе строк RandomStrings (п. 3.1). Мы использовали те же 2200 строк длиной 1000 символов (k_2=126), что и в предыдущем эксперименте п. А. Множество P было запомнено в LSH-лесе с помощью процедуры п. 2.2. 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. Вычислялось вышеописанное расстояние D_K (M(X'),M(Y')) (11) при |S|= 50, 200, 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. 3 для различных ограничений на максимальный размер возвращенного множества S и для следующих способов его дополнительной фильтрации:

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

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

Из рис. 4 видно, что 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 практике на небольшом значении.

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

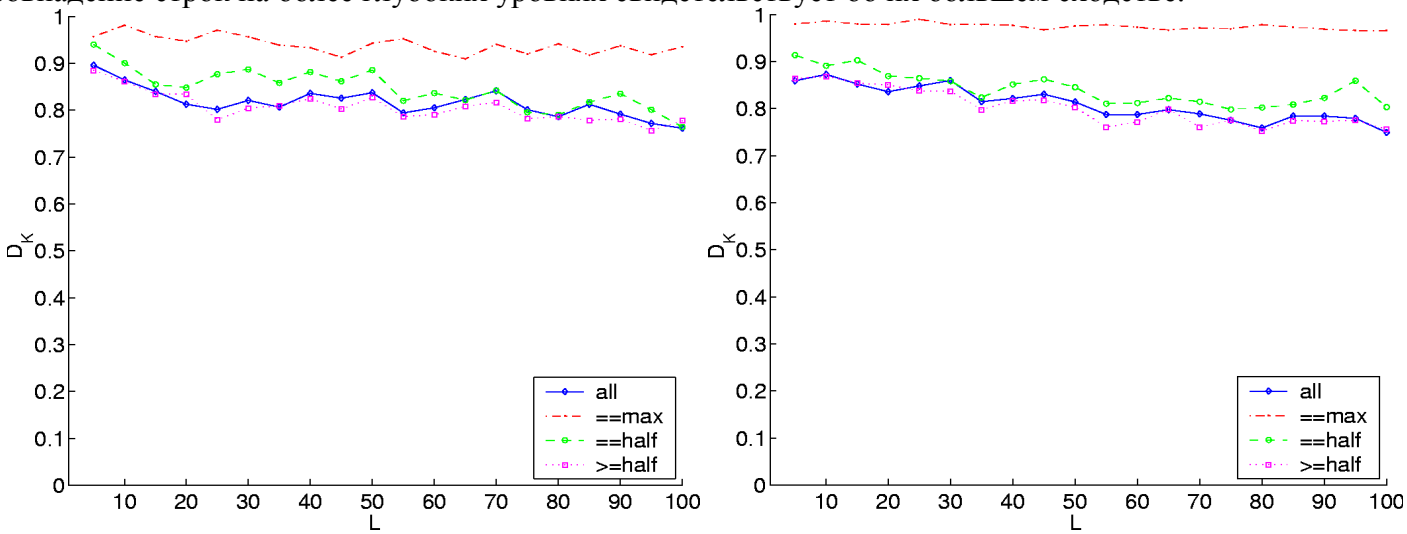


Рис. 3. Зависимость обобщенного коэффициента Кэндалла от количества деревьев L для размеров мультимножества S'= 50, 200.

4 Эксперименты с текстовыми коллекциями

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

4.1 Задача поиска дубликатов в текстовых коллекциях

Поиск (приближенных) дубликатов текстовых документов является операцией, которую необходимо эффективно выполнять в системах документооборота. Особенно критичной эта операция становится в поисковых машинах Интернет. 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.

Коллекции

Поиск дубликатов осуществлялся в следующих коллекциях новостных текстов Reuters-21578 [17] и учебных текстов British National Corpus [18], характеристики которых приведены в табл. 2.

Коллекция	**Максимальная длина текстов**	**Минимальная длина текстов**	**Медиана длины текстов**	**Всего текстов**
Reuters-21578	8316	13	519	21578
BNC	2494232	106	98250	4054

Табл. 2. Характеристики использованных текстовых коллекций (при применении С-функции isalpha())

Коллекция Reuters-21578 является стандартной коллекций 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 тексте, и т.п.), так и полных дубликатов.

Коллекция учебных текстов British National Corpus (BNC)– большая коллекция 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.

Методика поиска дубликатов

Для поиска дубликатов в текстовых коллекциях мы приняли, что дубликатами будут считаться строки, 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 символах 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.

Использовалась предварительная фильтрация текстов, оставляющая или только символы и цифры (С-функция isalnum()). 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.

Количество найденных дубликатов на коллекции Reuters-21578 и BNC

Найденное количество приближенных дубликатов в зависимости от значений K и n, для L = 1, 5 приведено в таблицах 3-4.

K \ n (L=1)	**100**	**150**	**250**	**500**	**750**	**1000**	**2000**	**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

K \ n (L=5)	**100**	**150**	**250**	**500**	**750**	**1000**	**2000**	**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	370	346	322	305	267	262	479	4823
200	369	346	322	305	267	261	264	2748

150	662	592	513	474	437	447	1253	5625
200	661	591	511	467	429	422	591	3499

Табл. 3. Количество найденных приближенных дубликатов в коллекции Reuters в зависимости от значений n и K, при L=1 и при L=5. Использовались символы C-функции isalnum().

K \ n (L=1)	100	150	250	500	750	1000	2000	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 \ n (L=5)	100	150	250	500	750	1000	2000	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	-

Табл. 4. Количество найденных приближенных дубликатов в коллекции BNC в зависимости от значений n и K, при L=1 и при L=5. Использовались символы C-функций isalnum().

Число приближенных дубликатов ожидаемо уменьшается при увеличении K, стабилизируясь при больших K на значениях, примерно соответствующих количеству дубликатов, найденных в работе [19] другим методом (320 дубликатов). 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 у хотя бы одного дерева. При "визуальной" проверке, однако, некоторые дубликаты оказались точными. Уменьшение количества дубликатов при увеличении длины обрезки для $K \geq 10$ при визуальной проверке показало, что оно вызвано мелкими опечатками в текстах. При меньших значения K эти опечатки могли не вызывать несовпадений хеш-векторов.

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 1.5 Гб памяти.

Сравнительные результаты поиска дубликатов

Результаты поиска дубликатов сравнивались с методом детерминированного вложения, описанным в [21]. За «золотой стандарт» было выбрано определение 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, большим или равным sim. Принимая поочередно за «настоящие дубликаты» 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). Результаты приведены на рис. 4.

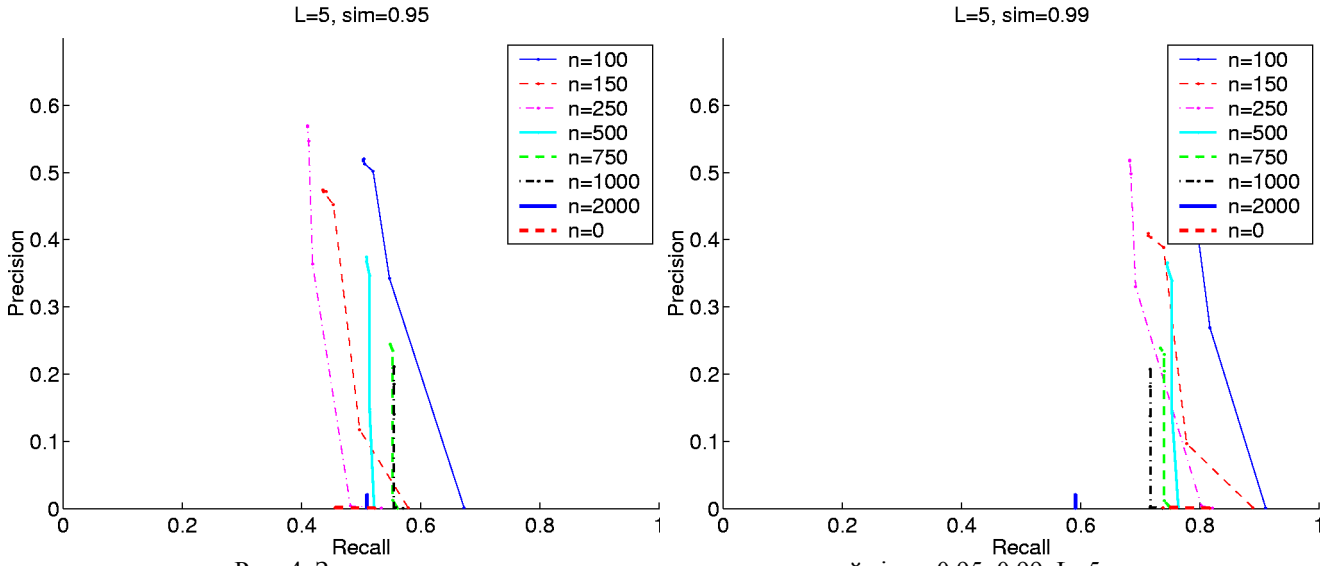
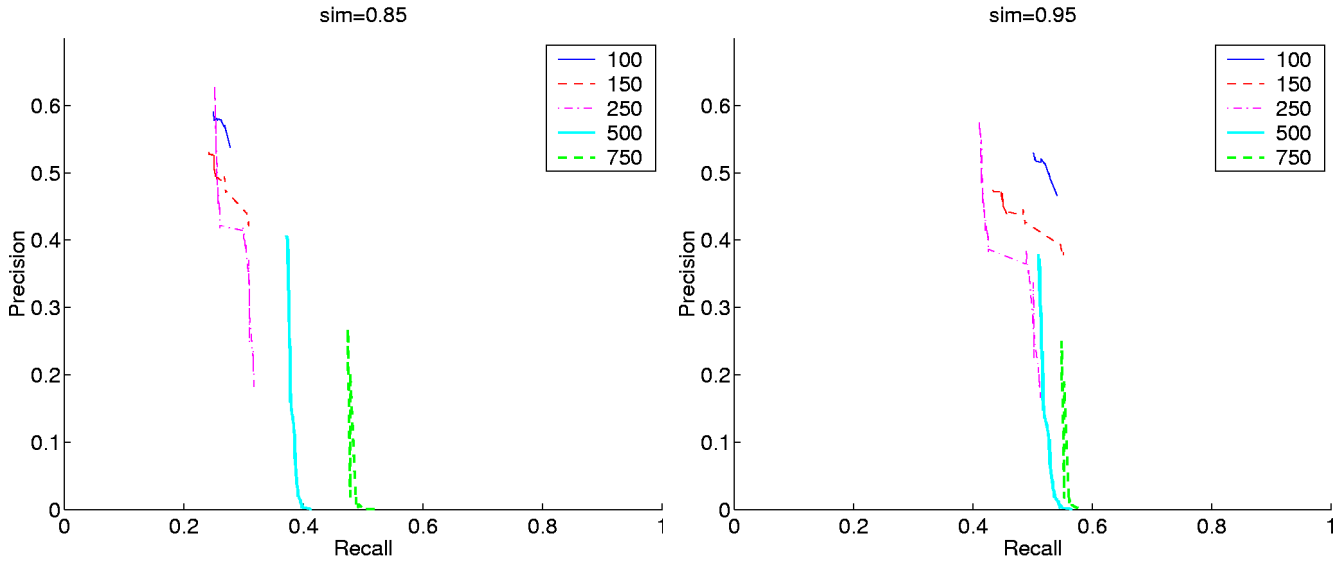


Рис. 4. Зависимость точности от полноты для значений sim = 0.95, 0.99. L=5

Как видно из рис. 4, при увеличении порога sim на значение функции String::Similarity полнота увеличивается, т.е. все 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.

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



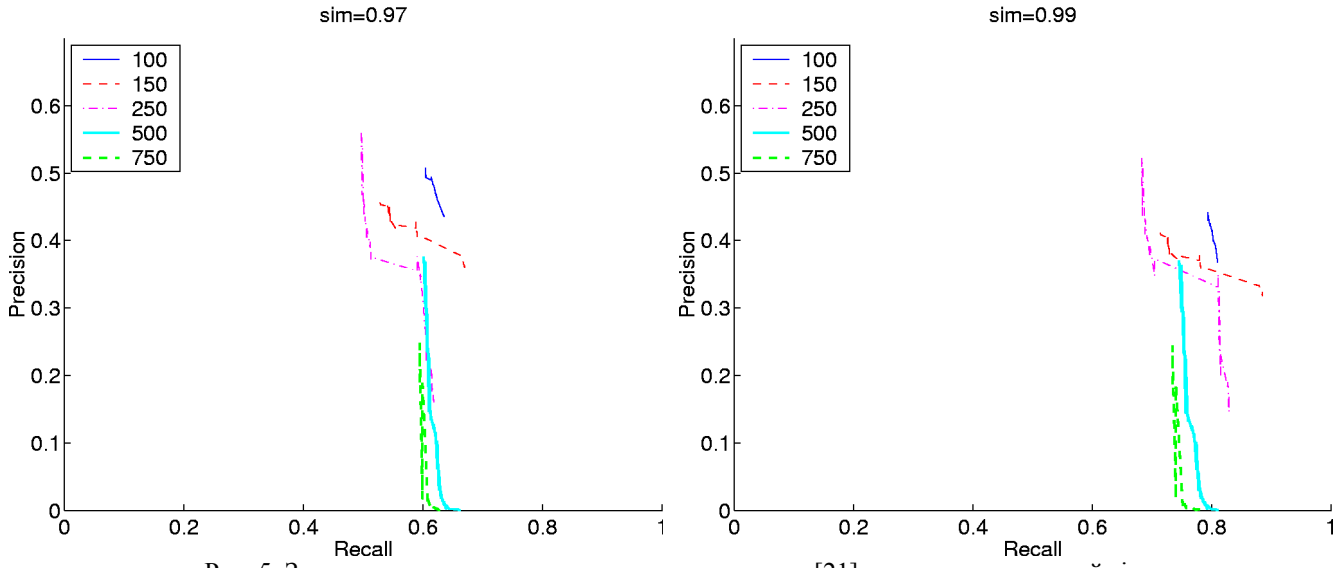


Рис. 5. Зависимость точности от полноты для метода [21] для разных значений sim.

Для сравнения пар значений точность-полнота использовалась интегральная оценка F-measure с $\alpha=1$, определяемая как $F_\alpha=(1+\alpha)rp/(\alpha p+r)$ [22], где r – полнота, p – точность. Для каждой из кривых, изображенных на рис. 5,6, было подсчитано максимальное значение F_{1max}, достигаемое на точках этой кривой. Полученные значения для n = 100, 250, 500, 750 изображены на рис 6.

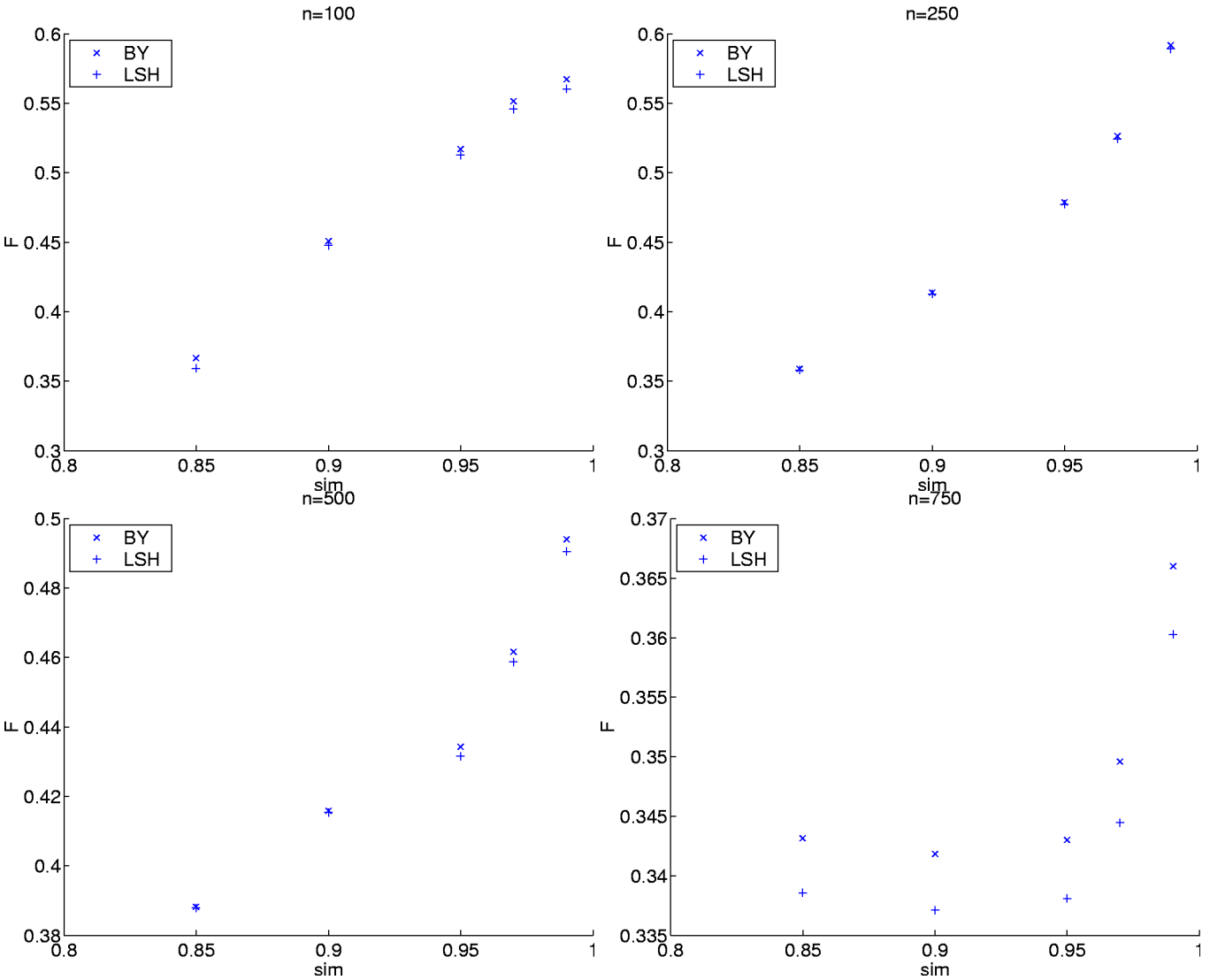


Рис. 6. Значения F_{1max} для n = 100, 250, 500, 750. Точки (крестики), обозначенные в легенде BY, соответствуют алгоритму из [21], плюсики – алгоритму поиска с помощью LSH-леса.

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 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 примерно равна длине запроса).

Были получены значения точности от 0.85 до 0.95 и полноты от 0.65 до 0.75, в зависимости от порога на значение функции PERL String::Similarity.

4.2 Задача оценки количества спама в коллекциях электронных писем

Обнаружение почтового спама (т.е. 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% [23] и увеличивается. 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.

Использованные коллекции
Были использованы тестовая база, широко применяющаяся разработчиками систем обнаружения спама для тестирования своих продуктов: TREC 2006 Spam Track [27]. Она содержит почтовые сообщения, размеченные экспертами на два класса: спам и не спам. Англоязычная часть коллекции TREC 2006 Spam Track содержит |P|=37822 размеченных реальных почтовых сообщений объемом 189 мегабайт, из которых 24912 (66%) – спам.

Схема экспериментов и оценка по *TREC*
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 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*%, 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.

Эксперименты и присвоение score

Предварительный фильтр оставлял в письмах одни лишь символы алфавита (C-фукнция 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_{II}, P (см. раздел 7).

На основании экспериментов раздела 4 мы выбрали два способа присвоения score письмам, по уровню в LSH-лес, к которому принадлежат приближенные ближайшие соседи к входныс письмам. А именно

- по максимальному уровню $k_{max} = K$;
- 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.

Результаты

Полученные ROC-кривые изображены на рис. 7, откуда видно, что при уровне *hm*%=5-10% успешно обнаруживается примерно 80% спама для Spam Track 2006.

Из проведенных экспериментов видно, что, реализовав идею обнаружения спама по приближенным дубликатам, можно отфильтровать значительную его часть, что позволяет говорить о применимости данного подхода в больших почтовых серверах, в качестве составной технологии (модуля) в более сложных системах. Чем централизованнее почтовый сервис и чем большее у него число пользователей, тем больший процент отфильтрованного спама можно ожидать.

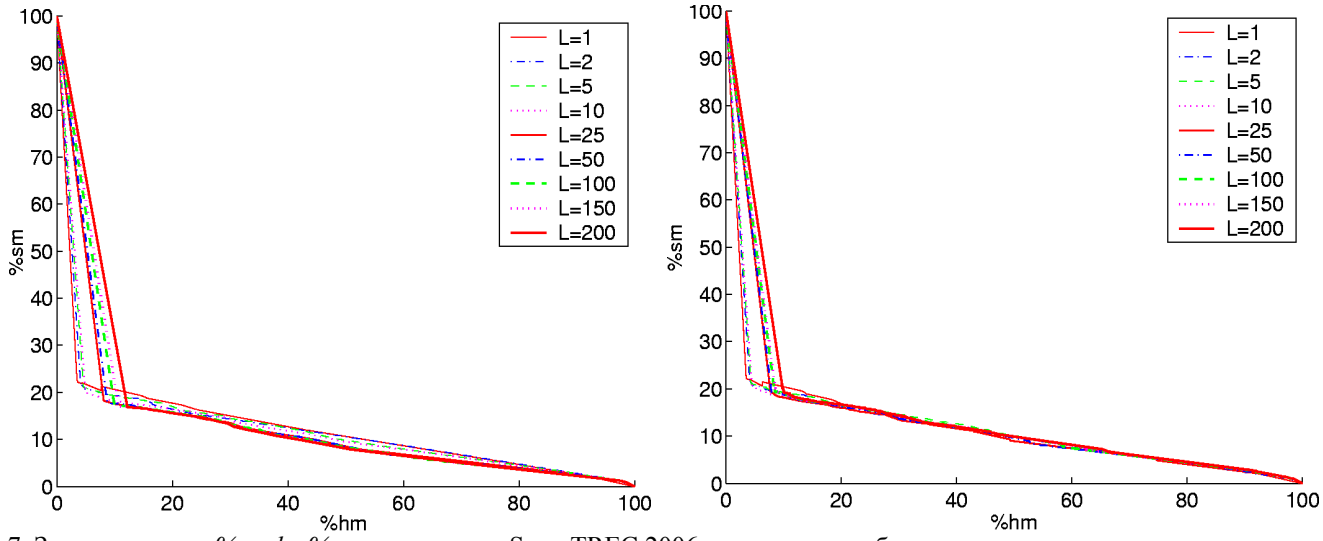


Рис. 7. Зависимость *sm%* от *hm%* для коллекции SpamTREC 2006 для двух способов присваивания score – по максимальному и по среднему уровню. Длина n=1000.

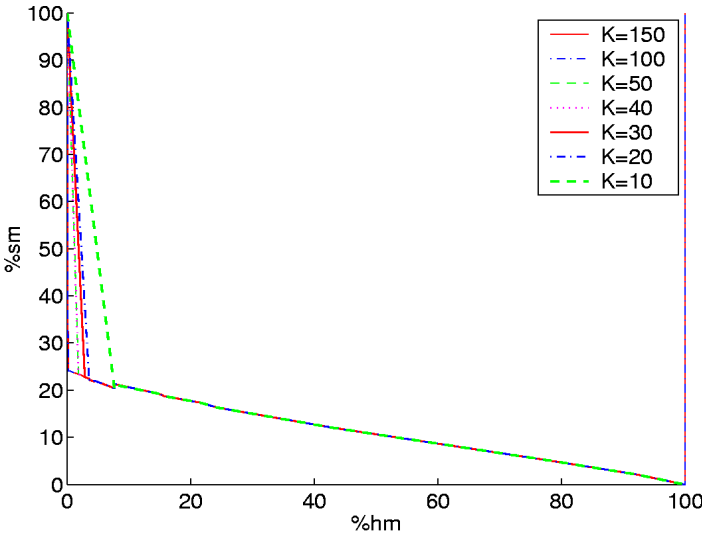


Рис. 8. Зависимость *sm%* от *hm%* для коллекции SpamTREC 2006 для способа присваивания score по максимальному уровню. L=1 и изменении K. Длина обрезки 1000 символов.

Из ROC-кривых на рис. 7 видно, что при L = 1 значения *hm*% часто получаются меньше, чем при L > 1. Это можно объяснить высокой степенью сходства части легальных писем с спамом, например, из-за использования html-формата, который мы оставляли в теле письма, а не убирали и/или не анализировали его код, как это принято в реальных системах обнаружения спама. Для L = 1 ROC-кривые для разных значений K приведены на рис. 8, откуда видно, что увеличение K приводит к более точной классификации легальных писем, так как улавливаются их более тонкие отличия от спама.

5 Выводы

Проведенные эксперименты подтвердили эффективность метода вложения расстояния редактирования в векторное пространство, а также основанного на нем рандомизированного, как методов нахождения приближенно ближайших строк [6].

Показана возможность применения рандомизированного метода в реальных практических задачах поиска дубликатов и обнаружения спама. Учитывая намеренное игнорирование при решении этих задач информации о специфике предметной области, которые обязательно должны 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 огромному 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 обнаружения спама Vipul, которая использует скетчи спам-сообщений, отмеченных как спам участниками системы. В качестве таких скетчей можно рассмотреть использование предлагаемых нами хеш-векторов.

В дальнейшем предполагается применить подхода 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.

6 Приложение. Значения параметров

В таблице 5 с проверочной целью приведены некоторые значения параметров процедуры LSH-лес, вычисляемые по значениям n и |P| для каждого эксперимента [6]. Во всех экспериментах были зафиксированы одинаковые значения k_l=1 и z=1.01.

Эксперимент	n	P	w=r	q_1	Δq	q_2	k_2	p_I	p_{II}	ρ	K	L
Точность и упорядоченность	1000	2200	100	67	8	75	126	0.712	0.708	0.984	20	1938
Reuters	100	21578	22	15	3	18	45	0.483	0.477	0.983	7	18256
	150	21578	29	20	3	23	50	0.555	0.537	0.948	9	12819
	250	21578	40	27	4	31	65	0.605	0.593	0.963	11	14935
	500	21578	63	43	6	49	94	0.657	0.648	0.971	15	16203
	750	21578	83	56	7	63	110	0.692	0.687	0.983	18	18151
	1000	21578	100	67	8	75	126	0.712	0.708	0.984	20	18306
	2000	21578	159	107	11	118	173	0.754	0.751	0.987	27	18938
	8316	21578	411	275	20	295	323	0.825	0.823	0.985	47	18546
BNC	100	4054	22	15	3	18	45	0.483	0.477	0.983	7	3527
	150	4054	29	20	3	23	50	0.555	0.537	0.948	9	2628
	250	4054	40	27	4	31	65	0.605	0.593	0.963	11	2984
	500	4054	63	43	6	49	94	0.657	0.648	0.971	15	3193
	750	4054	83	56	7	63	110	0.692	0.687	0.983	18	3510
	1000	4054	100	67	8	75	126	0.712	0.708	0.984	20	3535
	2000	4054	159	107	11	118	173	0.754	0.751	0.987	27	3636
	2494232	4054	18392	12262	153	12415	2951	0.963	0.963	0.990	389	3722
TREC 2006	1000	37822	100	67	8	75	126	0.712	0.708	0.984	20	31792
	5000	37822	293	196	16	212	256	0.804	0.802	0.987	39	33132

Табл. 5. Значения параметров процедуры LSH-лес в проведенных экпериментах

7 Литература

[1] Brin S., Davis J., García-Molina H. Copy detection mechanisms for digital documents. // Proc. SIGMOD. – 1995. – P. 398-409

[2] Gusfield D. Algorithms on Strings Trees and Sequences. – Cambridge Univeristy Press, 1997. – 532 p.

[3] Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР. – 1965. – Т. 163, – Вып. 4. – С. 845-848.

[4] Винцюк Т. К. Распознавание слов устной речи методами динамического программирования // Кибернетика.– 1968.– Вып. 1.– С. 81-88.

[5] Indyk P. Open problems // Workshop on Discrete Metric Spaces and their Algorithmic Applications / Ed. by Jirí Matoušek. – Haifa, 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 / Datar M., Immorlica N., Indyk P., Mirrokni V. // 20-th Annual Symposium on Computational Geometry. – 2004. – P. 253–262.

[9] Ukkonen E. Approximate string-matching with q-grams and maximal matches // Theor. Comput. Sci. – 92(3). – 1992. – 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 World Wide Web. – ACM Press, New York, 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 – P. 544

[14] Spink A., Bateman J., Jansen B. J. Searching the Web: Survey of EXCITE users // Internet Research: Electronic Networking Applications and Policy. – 9(4). – 1999. – 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 Journal 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 / Bar-Yossef Z., Jayram T. S., Krauthgamer R., Kumar R. // Proc. of the 45th Annual IEEE Symposium on Foundations of Computer Science, – IEEE, 2004 – P. 550-559
[22] van Rijsbergen C. J. Information Retireval. – Butterworths, London, 1979 –
[23] Messaging Anti-Abuse Working Group. “Email Metrics Program: The Network Operators' Perspective". Report N1 — 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 Conference 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 Conference. – NIST. – Gaithersburg, MD.