Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»
при Вычислительном центре им. А. А. Дородницына РАН

Аникеев Дмитрий Александрович

Эффективный поиск нечетких заимствований

03.03.01 — Прикладные математика и физика

Бакалаврская диссертация

Научный руководитель:
к.ф.-м.н.
Чехович Юрий Викторович

Москва
2018 г.

Содержание

Аннотация

Системы поиска совпадений в проверяемом текстовом документе на больших коллекциях эффективно отбирают небольшое количество кандидатов в источники заимствования. В данной работе исследуется задача поиска общих схожих подстрок в документе при сравнении с несколькими документами-кандидатами. Для решения задачи текстовому документу сопоставляется последовательность хешей обработанных слов. Алгоритмы четкого поиска позволяют найти короткие четкие дубликаты, которые распространяются вдоль текста с помощью вычисления расстояний Левенштейна между суффиксами и префиксами четкого дубликата. Исследуются оптимальные параметры модели, ее качество и быстродействие.

1 Введение

Цель работы заключается в эффективном решении задачи выявления схожих подстрок при сравнении с небольшой коллекцией возможных кандидатов. Выявление таких дубликатов необходимо при решении задачи обнаружения заимствований в текстовых документах [1, 2], сравнении последовательностей нуклеотидов и аминокислот [3], поиске других схожих строковых объектов. В данной работе рассматривается задача поиска нечетких заимствований в текстовых документах.

Актуальность задачи сравнения документа с небольшим количеством (обычно не более 20) кандидатов обусловлена тем, что задача отбора кандидатов в большой коллекции имеет эффективные решения. Такая задача решается методом поиска шинглов [4] или методами решения задачи информационного поиска [5].

Заимствованием считается копирование части кандидата в проверяемый документ. При сравнении двух текстов: проверяемого документа и кандидата невозможно установить, имел ли место факт заимствования, поэтому в качестве результата поиска заимствований выдаются нечеткие дубликаты — схожие общие подстроки в двух документах. Заимствования, порождающие нечеткие дубликаты можно условно разделить на две группы: дословные заимствования и парафраз.

Парафраз представляет собой копируемый текст, модифицированный с помощью таких приемов, как удаление, вставки и замена небольшой части слов, склейка и разбиение предложений (частичный парафраз). Более продвинутые приемы, такие как суммаризация, перестановка и замена большой части слов с сохранением смысла и комбинации предыдущих приемов (сильный парафраз) может быть сложно или практически невозможно обнаружить. Продвинутые модели анализа данных, использующие семантику языка справляются с этим, но при этом затраны с точки зрения времени и вычислительных ресурсов, а также выдают большое количество ложноположительных ответов на частичном парафразе.

Быстрые линейные модели, использующие суффиксное дерево [6] или суффиксный массив [7], обрабатывают документы порядка миллиона слов за секунду на обычных процессорах. Такие модели находят лишь четкие дубликаты и применяют простую предобработку, например склейку дубликатов.

В данной работе предлагается быстрая, линейная параметрическая модель обнаружения нечетких дубликатов. Первая фаза двухфазной модели находит короткие четкие дубликаты с помощью суффиксного массива или предложенным в статье ме-

тодом поиска шинглов. Вторая фаза распространяет найденный дубликаты вдоль текста, основываясь на расстоянии Левенштейна их суффиксов и префиксов.

Работы структрирована следующим образом. Во втором разделе приведена формальная постановка задачи. Третий раздел посвящен алгоритмам поиска четких дубликатов. В четвертом разделе предлагается модель поиска нечетких дубликатов. Вычислительный эксперимент определяет оптимальные параметры такой модели и приводит результаты анализа качества и быстродействия алгоритма.

2 Задача поиска дубликатов

В работе исследуется задача обнаружения заимствований. Сформулируем задачу для пары документов в терминах межстроковых расстояний.

2.1 Основные понятия и определения

Определение 2.1. *Последовательность* $Doc = [doc_1, doc_2, ..., doc_n]$ *называется проверяемым документом, а* $Src = [src_1, src_2, ..., src_m]$ *— источником. Символы* doc_i, src_i *принадлежат некоторому алфавиту* Σ.

Замечание 2.1. В качестве символа, минимального элемента документа, в данной работе рассматривается слово естественного языка или же его хеш. Например, для $CRC - 32$ преобразования $\Sigma = [0, 2^{32} - 1]$.

Определение 2.2. *Отображение* $Dup : \mathbb{N}^3 \mapsto \{0, 1\}$ *такое, что*

$$Dup(l_d, l_s, l) = 1 \Leftrightarrow doc_{l_d+i} = src_{l_s+i} \ \forall i \in [0, l-1]$$

называется индикатором множества четких дубликатов. Тройка чисел $\{l_d, l_s, l\}$ *называется четким дубликатом для пары документов* $\{doc, src\}$.

2.2 Формулировка задачи

В прикладных задачах необходимо выделить лишь те четкие дубликаты, которые не содержатся внутри других четких дубликатов, то есть те, которые нельзя расширить. Поэтому введем понятие множества максимальных четких дубликатов.

Определение 2.3. *Отображение* $MaxDup : \mathbb{N}^3 \mapsto \{0, 1\}$ *такое, что*

$$MaxDup(l_d, l_s, l) = 1 \Leftrightarrow \begin{cases} Dup(l_d, l_s, l) = 0 \\ Dup(l_d, l_s, l+1) = 1 \\ Dup(l_d - 1, l_s - 1, l+1) = 0 \end{cases}$$

называется индикатором множества максимальных четких дубликатов.

Определение 2.4. *Задача поиска максимальных четких дубликатов — это задача поиска множества*

$$Dups(Doc, Src) = \left\{ \{l_d, l_s, l\} = MaxDup^{-1}(1) \cap \{l > l_{\min}\} \right\}$$

Для формулировки задачи поиска нечетких дубликатов необходимо ввести функцию межстрокового расстояния $\rho : \Sigma^* \times \Sigma^* \mapsto \mathbb{R}^+$ определяющую близость двух строк.

Определение 2.5. *Отображение* $FuzzyDup : \mathbb{N}^4 \mapsto \{0, 1\}$ *такое, что*

$$FuzzyDup(l_d, l_s, l_1, l_2) = 1 \Leftrightarrow \rho\left(\{doc_i\}_{i=l_d}^{l_d+l_1}, \{src_i\}_{i=l_s}^{l_s+l_2}\right) \underbrace{\leq \varepsilon}_{\text{Внешний критерий}}$$

называется индикатором множества нечетких дубликатов. Четверка чисел $\{l_d, l_s, l_1, l_2\}$ *называется нечетким дубликатом для пары документов* $\{doc, src\}$.

Две подстроки, кодируемые началом и длиной являются нечетким дубликатом, если расстояние между этими подстроками меньше некоторого внешнего критерия, параметра модели ε.

Аналогично определяется множество максимальных нечетких дубликатов.

Определение 2.6. *Отображение* $MaxFuzzyDup : \mathbb{N}^4 \mapsto \{0, 1\}$ *такое, что*

$$MaxFuzzyDup(l_d, l_s, l_1, l_2) = 1 \Leftrightarrow \begin{cases} FuzzyDup(l_d, l_s, l_1, l_2) = 1 \\ FuzzyDup(l_d, l_s, l_1 + 1, l_2 + 1) = 0 \\ FuzzyDup(l_d, l_s, l_1 + 1, l_2) = 0 \\ FuzzyDup(l_d, l_s, l_1, l_2 + 1) = 0 \\ FuzzyDup(l_d - 1, l_s - 1, l_1 + 1, l_2 + 1) = 0 \\ FuzzyDup(l_d - 1, l_s, l_1 + 1, l_2) = 0 \\ FuzzyDup(l_d, l_s - 1, l_1, l_2 + 1) = 0 \end{cases}$$

называется индикатором множества максимальных нечетких дубликатов.

Замечание 2.2. Максимальность нечеткого дубликата понимается здесь в том смысле, что его нельзя расширить не более чем на 1 символ так, чтобы межстроковое расстояние не превысило пороговое значение.

Определение 2.7. *Задача поиска максимальных нечетких дубликатов — это задача поиска множества*

$$FuzzyDups(Doc, Src) = \left\{\{l_d, l_s, l_1, l_2\} = MaxFuzzyDup^{-1}(1) \cap \{l_1 > l_{\min}\} \cap \{l_2 > l_{\min}\}\right\},$$

где $l_{\min}$ *— параметр модели.*

2.3 Обучающая выборка

Определение 2.8. *Обучающая выборка задает отображение*

$$\mathfrak{D} = \left\{\{doc_i, src_i\} \mapsto \{TD_{i,j}\}_{j=1}^{n'}\right\}$$

которое сопоставляет каждой паре документ-источник список истинных заимствований. Задано отображение $Q(Dups, TD) : \{Dups_j\}_{j=1}^{n} \times \{Dups_{j'}\}_{j'=1}^{n'} \mapsto [0, 1]$ *вычисляющее близость разметок документов* $Dups$ *(обнаруженные моделью) и* TD *(размеченные экспертом).*

Замечание 2.3. В качестве множества дубликатов Dups выстапают множества $MaxDupes$ или $MaxFuzzyDupes$.

Для выполнения вычислительного эксперимента выбран набор данных для оценки методов поиска заимствований PlagEvalRus [**?**].

Нечеткие дубликаты обнаруженные для пары $\{doc_i, src_i\}$ обозначим

$$Dups_{i,j} = \{l_d, l_1, l_s, l_2\},\ j \in [1, n],$$

где l_d, l_s — индекс начала дубликата, а l_1, l_2 — его длина. Размером дубликата называется $|Dups_{i,j}| = l_d$. Определим метрики качества следующим образом.

Определение 2.9. *Микроусредненной точностью называется*

$$Q_{prec} = \frac{|\bigcup\limits_{i,j,i',j'} (Dups_{i,j} \sqcap TD_{i',j'})|}{|\bigcup\limits_{i',j'} TD_{i',j'}|}.$$

Определение 2.10. *Микроусредненной полнотой называется*

$$Q_{recall} = \frac{|\bigcup\limits_{i,j,i',j'}(Dups_{i,j} \sqcap TD_{i',j'})|}{|\bigcup\limits_{i,j} Dups_{i,j}|}.$$

Определение 2.11. *Гранулярностью разметки документов называется*

$$gran(Dups, TD) = \frac{1}{|Dups_{TD}|} \sum_{dup \in Dups_{TD}} |TD_{dup}|,$$

где $Dups_{TD} = \{Dups_j \in Dups : \quad \exists j' : TD'_j \cap Dups_j \neq \emptyset\}$, а
$TD_{dup} = \{td \in TD : \quad td \cap dup \neq \emptyset\}$.

Замечание 2.4. Иначе говоря, значение гранулярности равно усредненному количеству дубликатов, найденных внутри одного истинного заимствования.

Комбинированной оценкой близости разметок выступает

$$plagdet(Dups, TD) = \frac{F_\alpha}{\ln_2(1 + gran(Dups, TD))}.$$

В данной работе используется F_1-мера, поскольку нет оснований отдавать предпочтение точности либо полноте.

Для параметризованного алгоритма $A(\mathbf{w}, \mathfrak{D}) : (\Sigma^* \times \Sigma^*) \mapsto \{Dup_j\}_{j=1}^n$ оптимальные параметры модели находятся из условия

$$\hat{\mathbf{w}} = \arg\max_{\mathrm{w}} \mathrm{Q}(A(\mathbf{w}, \mathfrak{D}), TD). \tag{2.1}$$

3 Модель поиска дубликатов

3.1 Поиск четких дубликтов

Напомним, что в качестве минимального элемента текста, написанного на естественном языке, берется слово. Будем считать слова одинаковыми, если они совпадают посимвольно и различными в противном случае, вне зависимости от контекста и семантики. В качестве межстрокового расстояния возьмем количество замен, удалений и вставок слов, необходимых для приведения одной строки к другой. Тогда преобразование, которое сопоставляет каждому слову текстов его хеш-код сохраняет всю информацию о межстроковых расстояниях. Для CRC-32 хеш-функции $\Sigma = [0, 2^{32}-1]$. В данном разделе рассматриваются два алгоритма поиска четких дубликатов.

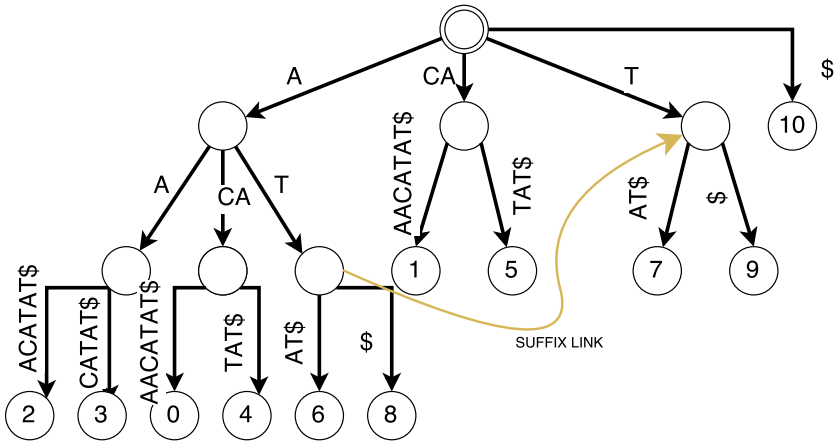

Рис. 1: Суффиксное дерево.

Суффиксный массив Для определения пересечений в последовательностях хешей используется структура, называемая суффиксным деревом. Суффиксное дерево — это дерево с помеченными строками, такое, что существует взаимно-однозначное соответствие между путями из корня дерева до его листа и множеством суффиксов строки. В конец строки добавляется терминальный символ, отличный от всех остальных, чтобы путь всегда заканчивался в листе. Поиск дубликатов в источнике по такой структуре осуществляется движением указателя по дереву при посимвольном чтении источника.

В статье [6] описан линейный алгоритм построения суффиксного дерева по документу и поиска четких дубликатов в источнике. В статье [7] приводится описание структуры, называемой суффиксным массивом, которая функционально эквивалентна суффиксному дереву, но имеет преимущества во времени работы и затратам памяти.

Теорема 3.1. *Ким, Парк (2007)*

Алгоритм Linearized Suffix Tree находит список всех четких дубликатов за $O((|doc| + |src|) \log |\Sigma|)$,

Поиск с помощью шинглов. Рассмотрим следующий алгоритм поиска четких дубликатов. Последовательность символов (хешей слов) фиксированной длины назовем шинглом. Двигая окно шингла вдоль документа составим хэш-таблицу, которая сопоставляет хешу шингла список его начал в документе. Далее, таким же образом двигаясь по источнику для каждого шингла найдем в хеш-таблице все его совпадения с источником, формируя список дубликатов. Затем, распространим все четкие

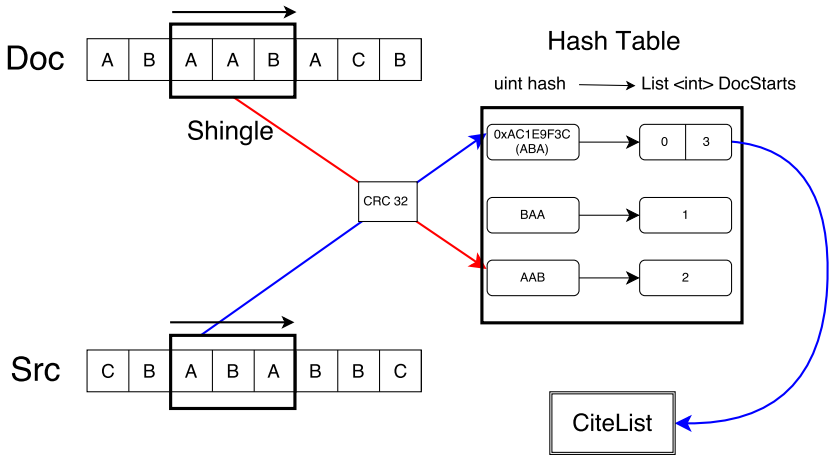

Рис. 2: Поиск четких дубликатов с помощью шинглов.

Алгоритм 3.1. Поиск четких дубликатов с помощью шинглов

Вход: $stringDoc[], stringSrc[]$;

Выход: `Четкие дубликаты`$Dup[]$;

1: Инициализировать по документу хеш-таблицу;
2: //(Шингл$\mapsto$ список позиций в документе с таким началом)
3: **для** каждого шингла в источнике
4: Найти значение в хеш-таблице;
5: Добавить все совпадения в Dup[];
6: Распространить четкие дубликаты.

дубликаты влево и вправо по документу, пока все символы совпадают.

Ширина окна является минимальной длиной дубликата. Будем называть этот параметр алгоритма MINLENGTH. Проверкой сложности процедур нетрудно убедиться в следующем.

Теорема 3.2. *Аникеев (2017)*

Поиск с помощью шинглов (3.1) *имеет сложность* $O(\sum_j |dup_j| + |doc| + |src|)$.

4 Модель нечеткого поиска

Гипотеза 4.1. *Внутри каждого нечеткого дубликата есть четкий дубликат длины MINLENGTH.*

Гипотеза 4.2. *(Гипотеза постоянного шума) Каждый нечеткий дубликат порожден заменой/удалением/вставкой слова в каждый промежуток четкого дубликата с*

фиксированной вероятностью

Если представить парафраз случайным преобразованием строки, в котором каждое слово изменяется/удаляется/вставляется в промежуток независимо и с фиксированной вероятностью, то верно следующее утверждение.

Лемма 4.1. *В условиях гипотезы порождения дубликатов* (4.2) *вероятность того, что внутри нечеткого дубликата есть четкий дубликат длины MINLENGTH стремится к единице при росте длины строки.*

Предлагается использовать двухфазный алгоритм, первой фазой которого является поиск четких дубликатов, выполненных одним из вышеперечисленных алгоритмов. Во второй фазе модель модифицирует найденный дубликаты.

4.1 Распространение дубликатов

Межстроковое расстояние Левенштейна – это количество удалений, замен и вставок, необходимых для приведения одной строки к другой. Относительное расстояние Левенштейна – это расстояние Левенштейна, поделенное на полусумму длин строк. В условиях гипотеза (4.2) количество модификаций пропорционально ее длине. Тогда относительное расстояние Левенштейна является адекватным критерием близости двух строк. Рассмотрим матрицу расстояний Левенштейна суффиксов найденного четкого дубликата.

$$D(i,j) = \begin{cases} i, & j = 0 \\ j, & i = 0 \\ min\{ & \\ \quad D(i,j-1)+1 & \\ \quad D(i-1,j)+1 & \texttt{иначе} \\ \quad D(i,j) + (int)(doc[i] != src[j]) & \\ \}, & \end{cases}$$

Расстояние Левенштейна в клетке зависит от значений в прилегающих клетках сверху, слева и слева-сверху. Пусть сначала ищутся строки с абсолютным расстоянием Левенштейна не больше заданного критерия, который будем называть LIMIT.

Расстояние между строками не меньше разности модулей длин этих строк, поэтому заполнять эту матрицу имеет смысл только вблизи диагонали (так называемое отсечение Укконена).

Под процедурой распространения будем понимать заполнение матрицы расстояний вблизи диагонали, пока все значения в столбце не станут больше $LIMIT$. После этого найдем вершину с наименьшим числом ошибок и будем двигать влево, вверх или влево вверх в вершину с наименьшим количеством ошибок пока не обнаружим совпадение последних символов суффикса. Назначим эти символы новыми правыми границами дубликата и проделаем то же самое с суффиксами. Такая процедура находит строки с абсолютным расстоянием не больше $2 \cdot LIMIT$.

	A	B	C	D	E	F	G	H
A	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
C	2	2	1	2	3	4	5	6
D	3	3	2	2	3	4	5	6
F	4	4	3	3	3	3	4	5
E	5	5	4	4	3	4	4	5
F	6	6	5	5	3	3	4	5
G	7	7	6	6	4	4	3	4

Рис. 3: Матрица расстояний Левенштейна.

Назовем центром нечеткого дубликата исходный четкий дубликат, относительно которого производилась процедура распространения. Чтобы ошибки не накапливались добавим в распространение процедуру, называемую сдвигом центра заимствования. Если при заполнении матрицы расстояний обнаружится совпадающая подстрока фиксированной длины EXPAND, то новой правой границей дубликата назовем конец этой подстроки и начнем распространение заново. Такая процедура позволит детектировать заимствования с малым относительным расстояния Левенштейна, но с большим абсолютным. Также она имеет преимущество перед процедурой распространения с ограничением на относительное расстояние Левенштейна: если нечеткий дубликат содержит несколько четких дубликатов, результаты распространения относительно каждого из разных центров дубликата будут совпадать.

Поскольку для каждого дубликата, благодаря отсечению Укконена, распространение происходит за O(длины префикса и суффикса) верно следующее утверждение.

Алгоритм 4.1. Поиск нечетких дубликатов с помощью шинглов

Вход: четкие дубликаты Dup[], string Doc[], string src[], LIMIT, EXPAND

Выход: нечеткие дубликаты FuzzyDup[]

```
1:  для каждого дубликата из Dup[]
2:    Limit = 0;
3:    пока Limit < LIMIT
4:      заполнить следующий столбец матрицы расстояний Левенштейна вблизи отсчения Укконена по реккурентной формуле;
5:      Limit = минимальное расстояние в столбце;
6:      если в столбце есть такие границы дубликата, что их суффиксы длины EXPAND совпадают то
7:        назначить эти границы новыми границами нечеткого дубликата;
8:        распространить границы пока новые префиксы и суффиксы совпадают;
9:        Limit = 0;
10:   для ячейки в столбце с наименьшим расстоянием Левенштейна
11:   пока последний символ в документе и источнике не совпадает
12:     //оптимальный путь
13:     перейти в смежную клетку по которой вычислялось значение в данной;
14:   добавить новый дубликат в FuzzyDup[]
```

Теорема 4.1. *Аникеев (2017)*

Процедура (4.1) *имеет сложность* $O(\sum_j |dup|_j)$.

4.2 Склейка дубликатов

В качестве постобработки найденного множества нечетких дубликатов рассматривается объединение или склейка нечетких дубликатов, близко расположенных в обоих документах.

Такая процедура позволяет во-первых, значительно снизить гранулярность, хоть и ценой точности. Во-вторых, таким образом увеличится частота детектирования модификация длинных последовательностей слов (удаление/вставка предложений и их частей, перестановка предложений местами), что приводит к увеличению полноты.

Расстоянием между дубликатами назовем максимум из расстояния между отрезками в источнике и отрезками в документе.

В процессе такой процедуры также удаляются все совпадающие дубликаты, по-

Алгоритм 4.2. Склейка дубликатов

Вход: нечеткие дубликаты FuzzyDup[], GLUE

Выход: нечеткие дубликаты GluedDup[]

```
1: отсортировать FuzzyDup[] по левой границе дубликата в документе, а в случае
   совпадения по убыванию длины в документе;
2: i=0;
3: Pointer = 0;
4: GluedDup=[];
5: пока  i< FuzzyDup.Length
6:    Prev=GlueDup[Pointer-1];
7:    New=FuzzyDup[i];
8:    если Prev не покрывает New то
9:       GluedDup[j+1] = New;
10:      Pointer++;
11:      пока Distance(Prev,New)<GLUE
12:         GluedDup[Pointer-1]=Glue(Prev,New);
13:         Pointer–;
14:         Prev = GluedDup[Pointer-1];
15:         New = GluedDup[Pointer];
16:   i++;
```

лученные при распространении множества дубликатов из разных четких дуликатов. Если при сортировке множества дубликатов сохранять лишь максимальные дубликаты, то такую сортировку можно сделать за $O(k + |doc|)$, где k - количество дубликатов. Поскольку указатель *Pointer* пробегает массив дубликатов лишь 1 раз, то верно следующее утверждение.

Теорема 4.2. *Аникеев (2018) Процедура* (4.2) *имеет сложность* $O(k + |doc|)$.

Выполняяя последовательно алгоритмы (3.1), (4.1), (4.2) получаем модель нечеткого поиска. Поскольку хеш-таблица, усеченные матрицы расстояний и массив дубликатов имеют затраты по памяти не более

$$O(\sum_j |dup_j| + |doc|)$$

из доказанных выше теорем вытекает следующее.

Теорема 4.3. *Аникеев (2018) Модель нечеткого поиска работает за* $O(\sum_j |dup_j| + |doc| + |src|)$.

5 Описание и предобработка данных

Выборка представляет собой размеченные пары текстов с синтезированными заимствованиями различных типов. Данные разбиты на 4 коллекции в зависимости от типа заимствований:

- Generated_copypast(4250 пар): Дословные заимствования, проверяемые документы сгенерированы машиной и содержат только четкие дубликаты.
- Generated_paraphrased(4250 пар): Средне модифицированные заимствования, сгенерированные машиной.
- Manually_paraphrased(713 пар): Дословные, средне и сильно модифицированные заимствования, сгенерированные человеком.
- Manually_paraphrased2(198 пар): Средне и сильно модифицированные заимствования, сгенерированные человеком.

Подробнее о правилах составления модифицированных текстов можно прочесть в статье [8].

Предобработка данных включает в себя удаление всех символов, кроме символов русского алфавита и приведение слов к нормальной форме (лемматизация). Затем, по документам строятся массивы хешей: каждое слово кодируется с помощью CRC-32 функции. Массив хешей содержит информацию об их разметке. После применения процедуры поиска по найденным дубликатам восстанавливаются их разметка в проверяемом документе и источнике. Все алгоритмы были реализованы на языке $C\sharp$.

6 Вычислительные эксперименты

В качестве базового алгоритма выбран суффиксный массив. Основной целью эксперимента являются сравнение скорости работы и качества модели нечеткого поиска с базовой моделью. Для этого необходимо провести оптимизацию параметров модели.

Таблица 1: Оптимальные параметры модели.

Параметр	Manually_Paraphrased	Manually_Paraphrased 2
MinCiteLength	5	4
LIMIT	5	6
EXPAND	2	1
GLUE	16	23

Таблица 2: Качество модели нечеткого поиска.

Метрика	Generated_Copy	Generated_Para	Manually_Para	Manually_Para 2
Precision	0.932	0.983	0.936	0.854
Recall	0.997	0.967	0.905	0.586
Granularity	1.001	1.001	1.008	1.013
PlagDet	0.961	0.780	0.922	0.551

Для поиска оптимальных параметров модели была выбрана коллеция Manually_Paraphrased, поскольку цель модели — обнаружить средне-модифицированные заимствования. В качестве метода оптимизации был выбран простой метод спуска с мультистартом (100 точек). Все старты сошлись в 1 точку. Модель содержит небольшое число дискретных параметров, поэтому можно не беспокоиться о переобучении. Оптимальные параметры описаны в таблице (1).

Видно, что параметры модели для коллекции с сильным парафразом допускают большее количество замен. В условиях оптимальных для коллекции Manually_Paraphrased параметров были определим метрики качества на каждой модели. Результаты представлены в таблице (2).

На рисунке (4) изображены срезы функций качества по различным параметрам модели в точке максимума на коллекции manually_paraphrased.

На рисунке (5) модель нечеткого поиска сравнивается с базовою моделью на различных колекциях. Предложенная модель превосходит суффиксный массив по качеству, в особенности на коллекциях ручного парафраза.

Для сравнения алгоритмов с более сложной моделью была выбрана модель, описанная в статье [10].

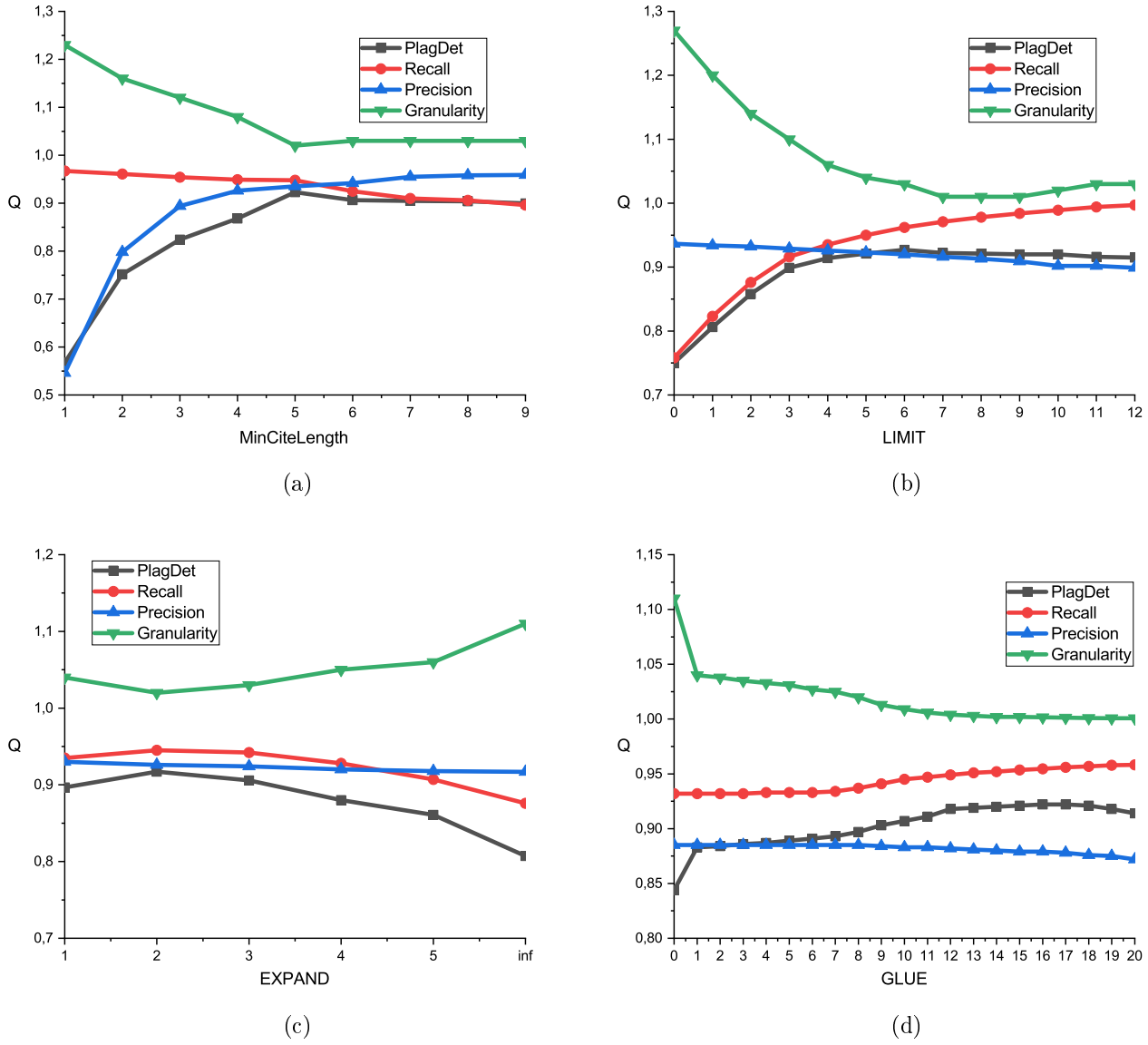

Рис. 4: Зависимость качества обнаружения заимствований от параметров модели.

Разметка PlagEvalRus содержит тип заимствования для каждого дубликата:

- CPY — дословное заимствование.
- ADD — добавление не более 30% слов.
- DEL — удаление не более 30% слов.
- CCT — разбиение предложения на 2.
- SSP — склейка предложений.
- LPR — небольшой парафраз, замена/удаление/вставка около 30% слов.

- HPR — сильный парафраз, замена/удаление/вставка более 30% слов.

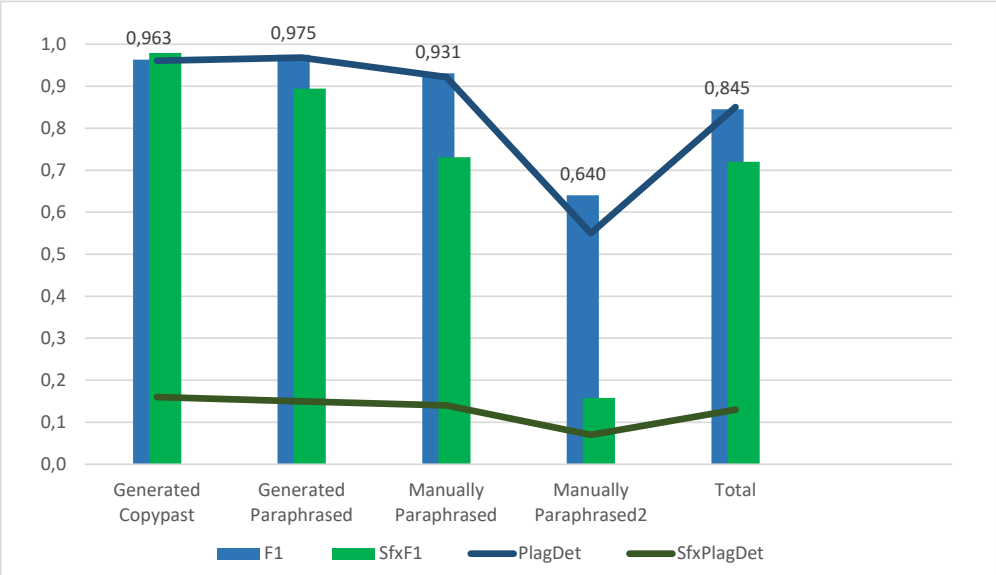

Рис. 5: Сравнение качества моделей на разных коллекциях.

Гранулярность модели нечеткого поиска и семантической модели менее 1.02 на любой коллекции. Полнота моделей в зависимости от типа заимствования указана в таблице (3).

Таблица 3: Зависимость полноты от типа заимствования

Par Type	SfxArray		FuzzySearch	SemanticSearch
Type	Recall	Granularity	Recall	Recall
CPY	98,6	13,1	99,5	94,6
ADD	61,1	9,57	96,4	93,1
DEL	63,6	10,9	95,9	92,8
CCT	88,2	17,0	98,8	94,4
SSP	87,2	19,4	98,5	95,9
LPR	35,6	7,67	94,1	94,0
HPR	11,5	6,24	62,3	79,3

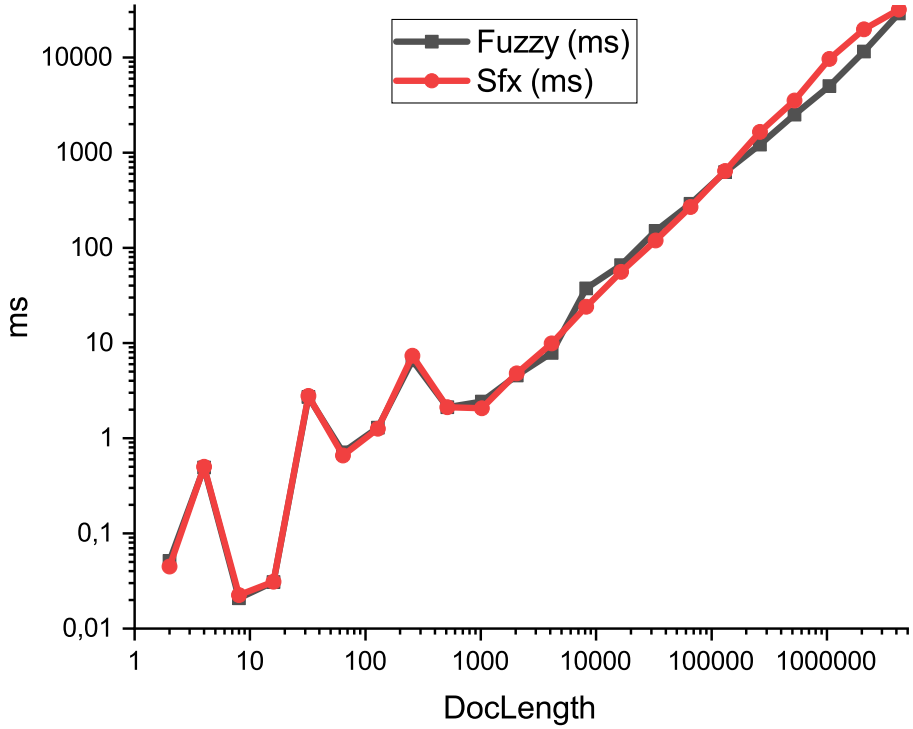

Рис. 6: Сравнение быстродействия моделей.

На рисунке (6) сравниваются времязатраты процедуры в зависимости от количества слов в документе. Обе модели показали схожий результат. Подтвердилась линейность модели нечеткого поиска.

7 Заключение

Предложенный алгоритм демонстрирует высокое качество при сравнении документа с кандидатом в источники заимствования, значительно превышающее качество базового алгоритма. Распространение цитаты значительно повышает полноту, а склейка дубликатов уменьшает гранулярность до близких к единице значений.

Основные результаты работы

- Предложен новый подход к поиску нечетких дубликатов при сравнении двух текстовых документов. Доказана линейность предложенной модели по сумме длин найденных дубликатов.

- Разработаны и реализованы описанные в работе алгоритмы.
- Проведены вычислительные эксперименты, демонстрирующие превосходство предложенной модели над базовой по качеству при небольших потерях времязатрат.

В заключении можно также перечислить предполагаемые направления дальнейших исследований. Использование семантического и контекстного анализа позволит увеличить качество обнаружения сильного парафраза. Необходимо изучить, имеется ли возможность включить такие методы в модель, не ухудшив сильно ее временную сложность.

Список литературы

[1] *Rakian S., Safi Esfahani F., Rastegari H.* A Persian Fuzzy Plagiarism Detection Scheme Based On Semantic Role Labeling // Journal Of Information Systems and Telecommunication, 2015. Vol. 3. P. 182–190.

[2] *Ezzikouri H., Erritali M., Oukessou M.* Fuzzy-Semantic Similarity for Automatic Multilingual Plagiarism Detection, 2017. Vol. 9. P. 86-90.

[3] *Капун Е.Д.* Разработка метода сравнения нуклеотидных последовательностей путем разбиения на фрагменты // Неопубликовано. Доступно: `http://is.ifmo.ru/bioinformatica/_kapun_thesis.pdf`

[4] *Lai C., Giuliani A., Semeraro G.* Emerging Ideas on Information Filtering and Retrieval: DART 2013. //Springer, 2018. 103 p.

[5] *Hagen M., Potthast M., Stein B.* Source Retrieval for Plagiarism Detection from Large Web Corpora: Recent Approaches // CLEF, 2015 Labs and Workshops, Notebook Papers. Toulouse, France.

[6] *Kim D. K., Kim M., Park H.* Linearized Suffix Tree: an Efficient Index Data Structure with the Capabilities of Suffix Trees and Suffix Arrays // Algorithmica, 2008. Vol. 52. P. 53-86.

[7] *Abouelhoda M. I, Kurtz S., Ohlebusch E.* Replacing suffix trrees with enhanced suffix arrays // Journal of Discrete Algorithms, 2004. Vol. 2. P. 53-86.

[8] *Sohenkov I. V., Zubare D. V., Smirnov I./, V.* The paraplag: russian dataset for paraphrased plagiarism detection. Computational Linguistics and Intellectual Technologies. // Papers from the Annual International Conference "Dialogue 2017. Vol. 1. P. 284–297.

[9] *Potthast M., Stein B., Barron-Cedeno A., Paolo R.* Evaluation Framework for Plagiarism Detection // 23rd International Conference on Computational Linguistics, COLING 2010. Bejing, China, 2010.

[10] *Hamza Osman A., Salim N., Salem M., Alteeb R., Abuobieda A.* An improved Plagiarism Detection Scheme Based On Semantic Role Labeling // Applied Soft Computing, 2012. Vol. 12. P. 1493–1502.

[11] *Аникеев Д. А., Чехович Ю. В.* Об эффективном алгоритме поиска нечетких заимствований // Информатика и ее применение. Подготовлено к печати.