УДК 004.012.82

И.В. ШОСТАК, И.В. ГРУЗДО

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

АНАЛИЗ ЭФФЕКТИВНОСТИ МЕТОДОВ ОБНАРУЖЕНИЯ ПЛАГИАТА В РАБОТАХ СТУДЕНТОВ ТЕХНИЧЕСКИХ СПЕЦИАЛЬНОСТЕЙ

В статье приведены результаты сравнительного анализа методов обнаружения фактов плагиата в естественно-языковых текстах в аспекте применения этих методов для выявления текстологических заимствований в работах студентов технических ВУЗов. На основе результатов проведенного анализа показана необходимость усовершенствования математического и методического обеспечения систем поиска плагиата, направленного на повышение эффективности таких систем. Показано, что в составе математического обеспечения систем поиска плагиата в студенческих работах целесообразно использовать алгоритм Бойера–Мура, при этом оправданным с практической точки зрения оказывается применение метода сопоставления строк.

Ключевые слова: *плагиат, студенческий плагиат, естественно-языковые тексты, информационно-поисковые системы.*

Введение

Общая характеристика проблемы. Главной задачей на сегодняшний день, стоящей перед преподавателями вузов Украины, является повышение качества образовательных услуг. Снижение уровня получаемого образования происходит за счет того, что при выполнении индивидуальных заданий и итоговых работ многие студенты руководствуются принципом быстрого и простого заимствования текстов из многочисленных источников, размещенных в Интернете: чужих рефератов, эссе, курсовых и дипломных проектов.

Указанные обстоятельства делают актуальным создание эффективного информационного инструментария для анализа естественно-языковых текстов в научных работах и как следствие, расширения возможностей мониторинга учебного процесса [1]. Решение данной задачи влечет за собой необходимость разработки эффективной системы анализа естественно-языковых текстов для обнаружения плагиата. Эффективность тестовых оценок зависит не только от вида текста и класса документов, но и от методов, которые применяются для сравнения и определения факта заимствования. Поэтому необходимо вначале проанализировать существующий математический аппарат, определить какие методы и алгоритмы используются при анализе естественно-языковых текстов.

В настоящее время в литературе известно сравнительно мало работ, посвященных современным методам поиска текстологических заимствований в аспекте их применимости для выявления заимствований в студенческих работах, что и определяет актуальность данной статьи. Поэтому существует необходимость анализа современного математического аппарата.

Цель статьи состоит в критическом обзоре методов обнаружения фактов плагиата при анализе естественно-языковых текстов в аспекте их применимости для выявления заимствований в работах студентов технических специальностей.

Постановка задачи исследования. Имеется множество методов обнаружения плагиата пригодных для выявления текстологических совпадений в различного рода документах.

Необходимо, сформулировав ряд характерных критериев, определить эффективность применения каждого из известных методов поиска текстологических совпадений для выявления заимствований в работах студентов гуманитарных специальностей с целью установления плагиата.

Решение указанной задачи предполагает реализацию следующей последовательности шагов:

– уточнить какая из полнотекстовых поисковых информационных систем относится к нашей задаче;

– рассмотреть основные типы поиска текстологических совпадений относящиеся к категории информационно-поисковых систем;

– провести критический анализ алгоритмов поиска для выявления текстологических совпадений относящиеся к методам сопоставления строк для выявления плагиата в различных типах студенческих работ.

Результатом решения задачи является статистическая информация о скорости, точности работы,

а так же количестве правильно найденных совпадений в анализируемом тексте.

Системы выборки данных и информационно-поисковые системы

В работе Рейзенберга [2] был представлен один из первых фундаментальных обзоров задач информационного поиска и информационно-поисковых систем, Information Retrieval", и, не смотря на то, что данная работа была опубликована в 1979 году, она содержит один из полных и исчерпывающих обзоров по информационному поиску на сегодняшний день. В приведённой классификации полнотекстовых поисковых информационных систем, в соответствии с работой Рейзенберга, существующие информационные системы, работающие с тестовыми документами, можно условно разделить на две категории: информационно-поисковые системы (в зарубежной терминологии они фигурирую под термином information retrieval systems, ИПС), и системы выборки данных (data retrieval systems, СВД). Стоит отметить, что данная классификация условна и в её контексте многие современные информационные системы совмещают в себе свойства, как систем выборки данных так и информационно-поисковых систем.

Базовые отличия систем выборки данных и информационно-поисковых систем, по Рейзенбергу, представлены в табл. 1.

Таблица 1

Сравнение систем выборки данных и информационно-поисковых систем

	СВД	ИПС
соответствие данных поисковому запросу	точное	частичное
классификация документов	детерминированная	вероятностная
язык запросов	искусственный	естественный
критерии выборки документов	булева функция релевантности	вероятностная функция релевантности
устойчивость к ошибкам в данных и запросах	неустойчивы	устойчивы

В данной работе проводится исследование систем, которые относятся к категории информационно-поисковых. В соответствии с этим рассмотрим основные типы поиска текстологических совпадений присущие данной категории.

Практически всю совокупность методов анализа текстовой информации можно отнести к категории информационно-поисковых систем. При этом их можно разделить на лингвистические методы и статистические (нелингвистические).

Лингвистические методы чаще всего ориентируются на извлечение смысла текста по его семантической структуре [3]. Нелингвистические методы не используют синтаксическую и семантическую информацию, заложенную в анализируемом тексте. Методы лингвистического анализа, включающие морфологический, синтаксический и семантический анализ, при сравнении смысловой схожести документов несомненно являются более предпочтительными, чем статистические.

Лингвистический анализ более точный и развернутый, вместе с тем он наиболее сложный и трудоемкий в использовании. Одним из классических примеров проблемной ситуации при анализе лингвистическим методом является омонимия, которая, в отличие от синонимии, не может быть решена построением дополнительного словаря [4]. При построении же полноценного морфологического анализатора возникают значительные затруднения с многообразием словоформ.

Так же следует отметить, что большинство лингвистических моделей ориентированы на один язык, и переход системы с английского на русский практически невозможен без существенной доработки всех модулей системы. Несмотря на значительные сложности при разработке лингвистических методов, работы в данном направлении ведутся уже многие годы и продолжаются в настоящее время.

Основные типы поиска текстологических совпадений используемые в информационно-поисковых системах

Булевый поиск. В данном типе информационного поиска, запросы строятся на основе элементарных терминов документов (слов или словоформ), находящихся между собой в отношениях, определённых предикатами, такими как дизъюнкция ($\vee$), конъюнкция ($\wedge$) и отрицание ($\neg$). Булевы запросы обычно используются в поиске на точное соответствие, в котором документы проверяются на наличие того или иного термина [5].

Поиск по релевантности. Предположим, что имеется некоторое множество электронных текстовых документов и существует пользователь, кото-

рый, взаимодействуя с системой и формулируя поисковые запросы, получает множество удовлетворяющих запросу документов. В этом случае задачей релевантного поиска является как можно более полная и точная выборка подмножества документов наиболее близких по смыслу (релевантных) поисковому запросу. Здесь под понятием близости документа поисковому запросу следует понимать некоторую функцию корреляции между каждым документом и запросом, вычисление которой позволяет определить степень близости информационного содержания документа и запроса.

Также широко применяются и методы статистического анализа текста документов [6]. Основной проблемой информационно-поисковых системах является автоматическая обработка и классификация документов для последующего вычисления функции релевантности, где в качестве языка запросов может использоваться язык запросов приближённый к естественному. Типичными представителями поисковых систем, осуществляющих релевантный поиск, являются системы Web поиска: Google, Яндекс, AstaVista. В таких системах, сначала осуществляется обычный булевый поиск (который может учитывать и грамматические формы терминов запроса) и на основе его результатов, производится поиск документов, которые имеют ссылки на документы из базового множества и наоборот.

Поиск по сходству. Представляет собой модификацию булевого поиска, где поисковой системой учитываются возможные неточности в задании поисковых терминов или в электронном представлении документов. С одной стороны поиск по сходству увеличивает объём выборки релевантных запросу документов, однако так же увеличивается и вероятность выборки документов реальная степень ролевантности которых очень мала. Поиск по сходству может быть очень полезен при выборке информации в специализированных электронных библиотеках таких как медицинские базы данных, или в системах распознавания текста, где существует определённая вероятность неточного задания термина в поисковом запросе или же в самом документе.

В настоящее время наиболее распространены поисковые системы, использующие два базовых принципа информационного поиска: поиск по ключевым словам (терминам) документов и поиск на основе кластерных методов и векторных моделей. Однако многие информационно-поисковые системы используют кластерные методы в качестве дополнения к поиску по ключевым словам, что позволяет существенно повысить эффективность и точность информационного поиска, а также эргономичность представления результатов поиска.

Следует отметить, что к основным обсуждаемым в литературе проблемам анализа строк относятся: сопоставление строк, вычисление расстояния между строками, нахождение самой длинной общей подпоследовательности, нечеткое сопоставление строк и поиск самой длинной повторяющейся подстроки. Ниже представлены постановка задач различные подходы к их решению.

Сопоставления строк обычно формулируется следующим образом [9]. Пусть заданы образец $x, |x| = m$, и текст $y, |y| = n$, где $m, n > 0$ и $m \le n$. Если x содержится в y, найти позицию первого вхождения x в y, то есть определить наименьшее i, при котором $y(i, i+m-1) = x(1,m)$. В обобщенной задаче требуется локализовать все вхождения x в y. Алгоритмы, решающие основную задачу, как правило, легко распространить на обобщенную. К наиболее «знаменитым» алгоритмам данной категории следует отнести: Кука, строки Фибоначчи, Кнута-Морриса, Бойера-Мура, Бойера-Мура-Хорспула, Бейза-Ейтс [9].

Расстояния между строками. Понятие функции, измеряющей расстояние, или метрики, используется в самых разных областях и часто используется для оценки сходства двух векторов. Такие функции применяют для определения силы связи двух признаков, или, например, в распознавании образов при сопоставлении шаблонов с различными частями изображений. Как известно, метрика задается функцией d с следующими свойствами [10]:

Неотрицательность $d(x,y) \ge 0 \forall x, y$.

Свойство нуля $d(x,y) \ge 0 \Leftrightarrow x = y$.

Симметричность $d(x,y) = d(x,y) \forall x, y$.

Неравенство треугольника $d(y,z) \le d(x,y) + d(y,z) \forall x, y, z$.

Последовательности длины n иногда объявляют векторами в Rn и применяют обычные расстояния: "шахматной доски" $\max i|x_i - y_i|$, "сити-блок" ("городских кварталов") $\sum_i |x_i - y_i|$ и Эвклида $\sqrt{\sum_i (x_i - y_i)^2}$.

При обработке строк, однако, символы не всегда уместно считать числами, несмотря на то, что коды символов являются ими. Кроме того, часто требуется сравнивать строки разной длины. Поэтому для сравнения строк обычно используют метрики, оценивающие максимальную стоимость преобразования одной строки в другую.

К наиболее «знаменитым» алгоритмам данной категории следует отнести: Расстояние Хемминга, расстоянием Левенштейна, Санкофа, Краскела и др.

Наибольшая общая подпоследовательность [11]. Подпоследовательность строки получается путем исключения из нее нуля или более символов, не обязательно смежных. Общая подпоследовательность двух строк – это строка, являющаяся подпоследовательностью каждой из них. Самая длинная из таких строк определяется, как, самая длинная общая подпоследовательность.

Итак, задача состоит в том, чтобы для данных строк x и y ($|x|, |y| > 0$) найти $|lcs(x, y)|$, где $lcs(x, y)$ – самая длинная общая подпоследовательность (longest common subsequence) x и y. Как правило, требуется найти не только длину $lcs(x, y)$, но и саму эту последовательность. К ним относятся методы Вагнера и Фишера, Хиршберга, Ханта и Мак-Илрой, Машека и Патерсона, Шиманского, Укконеном и др.

Нечеткое сопоставление строк. Обобщенная задача сопоставления строк, включающая в себя нахождение подстрок строки текста, близких к заданному образцу строки, называется также задачей нечеткого сопоставления строк [9]. Задачу нечеткого сопоставления строк можно сформулировать следующим образом:

Пусть даны образец x, $|x| = m$, и текст y, $|y| = n$, $m, n > 0$ и $m \le n$. Пусть даны также целое $k \ge 0$ и функция расстояния d. Требуется найти все подстроки s текста y такие, что $d(x, s) \le k$.

Задача, таким образом, состоит в том, чтобы при заданной функции расстояния найти все подстроки текста, отстоящие от образца не более, чем на k. Если d является расстоянием Хемминга, задача называется сопоставлением строк с k несовпадениями, если же d – расстояние Левенштейна, задача называется сопоставлением строк с k различиями (или, иногда, ошибками).

Исчерпывающий обзор алгоритмов нечеткого сопоставления строк представлен в работе Галила и Джанкарло [Galil, Giancarlo, 1988], сравнение сложности некоторых алгоритмов проведено также Гоннетом и Бейза-Ейтсом [12].

Максимальная повторяющаяся подстрока. Максимальная повторяющаяся подстрока – это подстрока, имеющая в данной строке не меньше двух различных вхождений, причем сопоставление нельзя продолжить ни в одном направлении.

Задача нахождения самой длинной подстроки (longest repeated substring), появляющейся в данной строке более одного раза, можно сформулировать следующим образом [9].

Для данной строки y, $|y| = n > 0$, найти самую длинную подстроку, встречающуюся в y больше одного раза. Самая длинная повторяющаяся подстрока – это самая длинная из строк максимальной длины x, такая что $y = uxvxw$ для некоторых строк u, v и w, где $|u| \ge 0$, $|v| \ge 0$ и $|w| \ge 0$. Однако, самую длинную повторяющуюся подстроку можно найти за линейное время, если использовать одну из структур данных, предложенных Вейнером (Weiner, 1973), содержащих компактный индекс для всех различных подстрок строки.

Проанализировав основные задачи анализа строк можно сделать вывод, что наиболее полно проблематику нашей области исследования затрагивают задачи сопоставления строк, а так же максимальная повторяющаяся подстрока.

Рассмотрим алгоритмы поиска для выявления текстологических совпадений относящиеся к методам сопоставления строк более подробно.

Алгоритмы поиска для выявления текстологических совпадений относящиеся к методам сопоставления строк

Алгоритм последовательного (прямого) поиска (The Brute Force Algorithm) [10, 12].

Обозначим S – слово, в котором ищется образец x. Пусть m и n – длины слов S и x соответственно. Можно сравнить со словом x все подслова S, которые начинаются с позиций $1, 2, ..., m - n + 1$ в слове S, в случае равенства выводится соответствующая позиция [13, 14].

Максимальное, количество сравнений будет равно $O((m - n + 1) \times n + 1)$. Например, найдя строку aabc и обнаружив несоответствие в четвертом символе (совпало только aab), алгоритм будет продолжать сравнивать строку, начиная со следующего символа, хотя это однозначно не приведет к результату.

Следующий метод работает намного быстрее простейшего, но, к сожалению, накладывает некоторые ограничения на текст и искомую строку.

Алгоритм Рабина представляет собой модификацию линейного алгоритма. Он основан на следующей идее [15], в слове A, длина которого равна m, мы ищем образец x длины n. Вырежем "окошечко" размером n и передвигаем его по входному слову. Нас интересует, не совпадает ли слово в "окошечке" с заданным образцом.

Вместо побуквенного сравнения фиксируем некоторую числовую функцию на словах длины n, тогда задача сводится к сравнению чисел, что, намного быстрее. Если значения этой функции на слове в "окошечке" и на образце различны, то совпаде-

ния нет. Только если значения одинаковы, необходимо проверять последовательно совпадение по буквам».

Этот алгоритм выполняет линейный проход по строке (n шагов) и линейный проход по всему тексту (m шагов), стало быть, общее время работы есть $O(n+m)$. При этом не учитывается временная сложность вычисления хеш-функции. Тогда, время работы алгоритма линейно зависит от размера строки и текста, стало быть, программа работает быстро.

Строго говоря, время работы всего алгоритма в целом, есть $O(n+m+mn/P)$, mn/P достаточно невелико, так что сложность работы почти линейная.

Алгоритм Рабина и алгоритм последовательного поиска являются алгоритмами с наименьшими трудозатратами, поэтому они годятся для использования при решении некоторого класса задач. Однако эти алгоритмы не являются наиболее оптимальными (хотя бы потому, что иногда выполняют явно бесполезную работу, о чем было сказано выше), поэтому перейдём к следующему классу алгоритмов. Эти алгоритмы появились в результате тщательного исследования алгоритма последовательного поиска.

Рассмотрим **алгоритм Кнута – Морриса – Пратта** – алгоритм поиска образца (подстроки) в строке. Алгоритм был открыт Д. Кнутом и В. Праттом и, независимо от них, Д. Моррисом[16]. Результаты своей работы они опубликовали совместно в 1977 году [17].

Поставим следующую задачу: имеется образец S и строка T, и нужно определить индекс, начиная с которого строка S содержится в строке T. Если S не содержится в T – вернуть индекс, который не может быть интерпретирован как позиция в строке (например, отрицательное число). При необходимости отслеживать каждое вхождение образца в текст имеет смысл завести дополнительную функцию, вызываемую при каждом обнаружении образца.

Рассмотрим сравнение строк на позиции i, где образец $S[0, m-1]$ сопоставляется с частью текста $T[i, i+m-1]$. Предположим, что первое несовпадение произошло между $T[i+j]$ и $S[j]$, где $1<j<m$. Тогда $T[i, i+j-1]=S[0, j-1]=P$ и $a=T[i+j]\neq S[j]=b$.

При сдвиге вполне можно ожидать, что префикс (начальные символы) образца S сойдется с какимнибудь суффиксом (подсловом) текста P. Длина наиболее длинного префикса, являющегося одновременно суффиксом есть префикс-функция от строки P.

Это приводит нас к следующему алгоритму: пусть $\pi[j]$ – префикс-функция от строки $S[0, j-1]$. Тогда после сдвига мы можем возобновить сравнения с места $T[i+j]$ и $S[\pi[j]]$ без потери возможного местонахождения образца. Средствами амортизацононного анализа можно показать, что таблица π может быть вычислена за $\Theta(m)$ сравнений перед началом поиска. А поскольку строка T будет пройдена ровно один раз, суммарное время работы алгоритма будет равно $\Theta(m+n)$, где n – длина текста T.

Пример практической реализации – пусть ищется строка S_1 в строке S_2. Построим строку $S=S_1\$S_2$, где символ \$ – символ, не встречающийся ни в S_1, ни в S_2. Далее вычислим значения префикс-функции от строки S и всех её префиксов. Теперь, если префикс-функция от префикса строки S длины i равна n, где n – длина S_1, и $i>n$, то в строке S_2 есть вхождение S_1, начиная с позиции $i-2n$. Заметим, что алгоритм последовательного поиска и алгоритм КМП помимо нахождения самих строк считают, сколько символов совпало в процессе работы.

Алгоритм Бойера-Мура, разработанный двумя учеными – Бойером (Robert S. Boyer) и Муром (Strother Moore), считается наиболее быстрым среди алгоритмов общего назначения, предназначенных для поиска подстроки в строке.

Простейший вариант алгоритма Бойера-Мура состоит из следующих шагов. На первом шаге строим таблицу смещений для искомого образца. Далее совмещаем начало строки и образца и начинаем проверку с последнего символа образца. Если последний символ образца и соответствующий ему при наложении символ строки не совпадают, образец сдвигается относительно строки на величину, полученную из таблицы смещений, и снова проводится сравнение, начиная с последнего символа образца. Если же символы совпадают, производится сравнение предпоследнего символа образца и т.д. Если все символы образца совпали с наложенными символами строки, значит, была найдена подстрока и поиск окончен. Если же какой-то (не последний) символ образца не совпадает с соответствующим символом строки, мы сдвигаем образец на один символ вправо и снова начинаем проверку с последнего символа. Весь алгоритм выполняется до тех пор, пока либо не будет найдено вхождение искомого образца, либо не будет достигнут конец строки.

Величина сдвига в случае несовпадения последнего символа вычисляется исходя из следующих соображений: сдвиг образца должен быть минимальным, таким, чтобы не пропустить вхождение образца в строке. Если данный символ строки встречается в образце, мы смещаем образец таким образом, чтобы символ строки совпал с самым пра-

вым вхождением этого символа в образце. Если же образец вообще не содержит этого символа, мы сдвигаем образец на величину, равную его длине, так что первый символ образца накладывается на следующий за проверявшимся символ строки. Величина смещения для каждого символа образца зависит только от порядка символов в образце, поэтому смещения удобно вычислить заранее и хранить в виде одномерного массива, где каждому символу алфавита соответствует смещение относительно последнего символа образца. Графически все это можно изобразить как представлено на рис. 1.

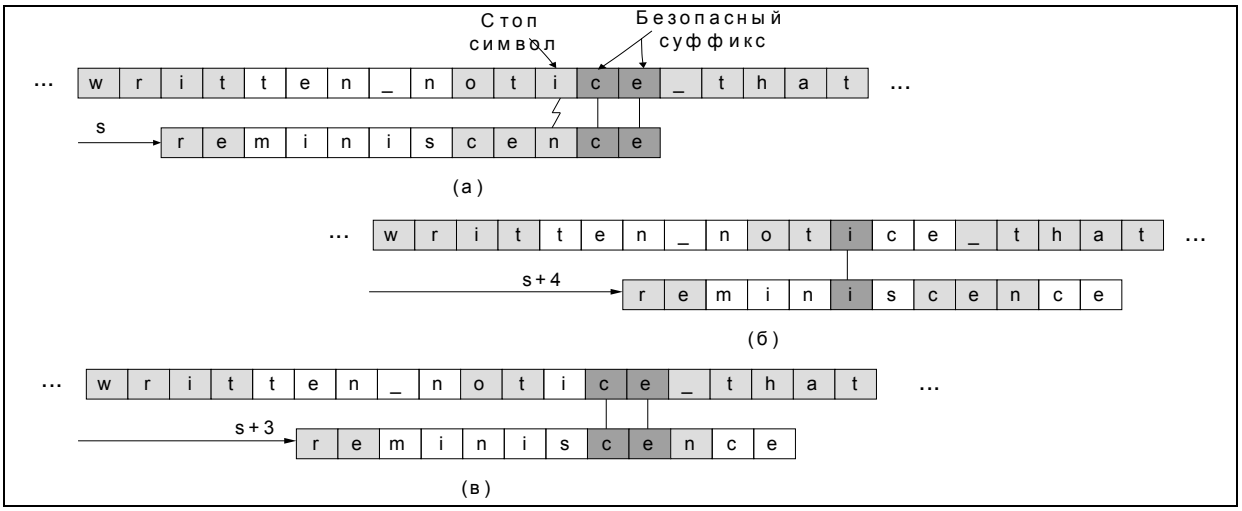

Рис. 1. Графическая демонстрация алгоритма Бойера – Мура

Поиск подстрок с помощью конечного автомата. По определению, конечный автомат представляет собой пятерку $M = (Q, q0, A, \Sigma, \delta)$, где Q – конечное множество состояний; $q0 \in Q$ – начальное состояние; $A \in Q$ – конечное множество допускающих состояний; Σ – конечный входной алфавит; δ – функция $Q \times \Sigma \to Q$, называемая функцией переходов автомата.Первоначально конечный автомат находится в состоянии $q0$. Затем он по очереди читает символы из входной строки. Находясь в состоянии q и читая символ *a*, автомат переходит в состояние $\delta(q, a)$. Если автомат находится в состоянии $q \in A$ то он допускает прочитанную часть входной строки. Если $q \notin A$, то прочитанная часть строки отвергнута.

С конечным состоянием M связана функция $\varphi : \Sigma^* \to Q$, называемая функцией конечного состояния, определяемая следующим образом: $\varphi(w)$ есть состояние, в которое придет автомат (из начального состояния), прочитав строку w. Автомат допускает строку w тогда и только тогда, когда $\varphi(w) \in A$. Для каждого образца P можно построить конечный автомат, ищущий этот образец в тексте. Первым шагом в построении автомата, соответствующего строке-образцу $P[1..m]$, является построение по P вспомогательной суффикс-функциии (как в алгоритме КМП).

Определим конечный автомат, соответствующий образцу $P[1..m]$, следующим образом:

– его множество состояний. Начальное состояние $q0 = 0$. Единственное допускающее состояние m;

– функция переходов δ определена соотношением (q – состояние, $a \in \Sigma$ – символ): $\delta(q,a) = \delta(Pqa)$.

Поясним это соотношение. Требуется сконструировать автомат таким образом, чтобы при его действии на строку T соотношение $\varphi(Ti) = \delta(Ti)$ являлось инвариантом (тогда равенство $\varphi(Ti) = m$ будет равносильно тому, что P входит в T со сдвигом i – m, и автомат тем самым найдет все допустимые сдвиги).

Задача поиска подстроки состоит из 2-х частей:

– построение автомата по образцу (определение функции переходов для заданного образца);

– применение этого автомата для поиска вхождений образца в заданный текст.

Построим по граф переходов автомата (рис. 6), распознающего образец abababaca.

Находясь в состоянии q и прочитав очередной символ a, автомат переходит в состояние $\delta(q,a)$. Обратим внимание, что его остов помечен символами образца (эти переходы выделены жирными стрелками). Жирные стрелки, идущие слева направо, соответствуют успешным этапам поиска подстроки P – следующий входной символ совпадает с очередным символом образца. Стрелки, идущие справа налево, соответствуют неудачам – следующий входной символ отличается от очередного символа образца.

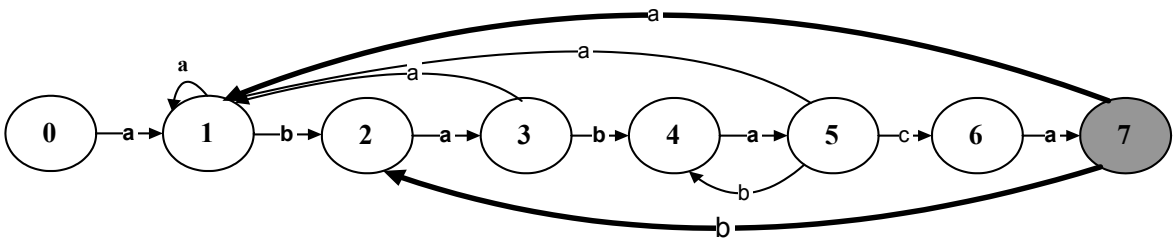

Рис. 2. Граф переходов автомата:
0 – исходное состояние; 7 – единственное допускающее состояние (зачернено).
Если из вершины i в вершину j ведет стрелка, помеченная буквой a, то это означает, что $\delta(i,a) = j$

Проанализировав теоретические данные по данным алгоритмам сведем их в общую таблицу которая будет демонстрировать зависимость алгоритма, от таких показателей: время на обработку, среднее время необходимое на разработку, худшее время поиска и затраты памяти (табл. 2). .

Таблица 2

Анализ алгоритмов поиска

Алгоритм	Время на пред. обработку	Среднее время поиска	Худшее время поиска	Затраты памяти	Примечания
Алгоритм прямого поиска	Нет	O((m–n+1)*n+1)/2	O((m–n+1)*n+1)	Нет	Малые трудозатраты на программу, малая эффективность.
Алгоритм Рабина	Нет	O(m+n)	O((m-n+1)*n+1)	Нет	
КМП	O(m)	O(n+m)	O(n+m)	O(m)	Универсальный алгоритм, если неизвестна длина образца
БМ	O(m+s)	O(n+m)	O(n*m)	O(m+s)	Алгоритмы этой группы наиболее эффективны в обычных ситуациях. Быстродействие повышается при увеличении образца или алфавита.

Исходя из полученных результатов, видно, что алгоритм Бойера – Мура является ведущим по всем параметрам, казалось бы, найден самый эффективный алгоритм. Но не всегда результаты доказанные в теории подтверждают себя на практике.Поэтому не возможно сделать вывод, какой из алгоритмов является самым оптимальным и для какого класса задач наиболее эффективен тот или иной алгоритм, так же следует отметить то, что имеются различные узконаправленные улучшения, для каждого алгоритма учитывающие при этом специфику области. Но при этом необходимо отметить то, что алгоритм поиска подстроки в строке следует выбирать только после точной постановки задачи, которые должна выполнять программа

Но это все хорошо в теории, а как выглядит на практике неизвестно, поскольку никто не проводил комплексного анализа алгоритмов между собой даже на одной практической задаче. Не всегда теоретические постулаты работают так как предполагалось в теории поэтому для того, чтобы более полно судить об эффективности каждого алгоритма необходимо его проверить и с практической точки зрения. При этом целесообразно продолжить исследование алгоритмов, относящихся к методам анализа максимально повторяющаяся подстроки.

Выводы

На данном этапе актуальной является проблема повышения уровня образования и, как следствие, создания эффективного информационного инструментария для анализа естественно-языковых текстов в студенческих работах и расширения функций мониторинга учебного процесса. В связи с высокой стоимостью и сложностью разработки таких программ, необходимо обосновать выбор эффективного математического аппарата для будущей системы.

Работа посвящена исследованию таких систем, которые относятся к категории информационно-поисковые системы. Рассмотрены методы поиска текстологических совпадений присущие данной категории это лингвистические и статистические (внелингвистические) методы. Лингвистические методы чаще всего ориентируются на извлечение смысла текста по его семантической структуре. Нелингвис-

тические методы не используют синтаксическую и семантическую информацию, заложенную в анализируемом тексте. Методы лингвистического анализа, включающие морфологический, синтаксический и семантический анализ, при сравнении смысловой схожести документов несомненно являются более предпочтительными, чем статистические.

Большинство лингвистических моделей ориентированы на один язык, и переход системы с английского на русский практически невозможен без существенной доработки всех модулей системы. Несмотря на значительные сложности при разработке лингвистических методов, работы в данном направлении ведутся многие годы и продолжаются в настоящее время.

Наиболее распространены поисковые системы, использующие два базовых принципа информационного поиска: поиск по ключевым словам документов и поиск на основе кластерных методов и векторных моделей. Однако многие информационно-поисковые системы используют кластерные методы в качестве дополнения к поиску по ключевым словам, что позволяет существенно повысить эффективность и точность информационного поиска, а также удобочитаемость результатов поиска. К системам, использующим поиск по ключевым словам можно отнести большинство широко распространенных поисковых систем, среди них такие системы Web поиска как: Яндекс, Google, AstaVista, Yahoo.

К основным, обсуждаемым в литературе проблемам анализа строк относятся: сопоставление строк, вычисление расстояния между строками, нахождение самой длинной общей подпоследовательности, нечеткое сопоставление строк и поиск самой длинной повторяющейся подстроки. Проанализировав основные задачи анализа строк следует отметить, что наиболее полно специфику области исследования затрагивают задачи сопоставления строк, а так же максимальная повторяющаяся подстрока.

Проведен анализ методов и их алгоритмов в частности алгоритмов (прямого поиска, алгоритм Рабина, Кнута–Морриса–Пратта, Бойера–Мура). Анализ показал, что для выявления текстологических совпадений в работах относящиеся к методам сопоставления, алгоритм Бойера–Мура является ведущим по всем параметрам, казалось бы, найден самый эффективный алгоритм. Но не всегда результаты доказанные в теории подтверждают себя на практике. Поэтому невозможно сделать вывод, какой из алгоритмов является самым оптимальным и для какого класса задач наиболее эффективен тот или иной алгоритм, так же следует отметить то, что имеется возможность различных узконаправленных улучшений, для каждого алгоритма учитывающих при этом специфику предметной области. При этом необходимо отметить, что алгоритм поиска подстроки в строке следует выбирать только после точной постановки задачи, которую должна выполнять программа.

В частности для применения данных алгоритмов, в практических задачах, где возможно появление отклонений, существует необходимость проведения эксперимента, для более полной оценки каждого алгоритма. Так же необходимо продолжить исследование алгоритмов, относящихся к методам анализа максимально повторяющейся подстроки.

Литература

1. Груздо И.В. Проблемы машинного анализа естественно-языковых текстов для обнаружения плагиата в научных работах», / И.В. Груздо // Інтегровані комп'ютерні технології в машинобудуванні ІКТМ - 2010: Всеукраїнська науково-технічна конференція, Харків НАУ "ХАІ", 22-24 ноября 2010 г., Т 3. – 220 с.

2. Van Rijsbergen C.J. Information Retrieval. Dept. of Computer Science / C.J. van Rijsbergen. – University of Glasgow, 1979.

3. Чугреев В.Л. Модель структурного представления текстовой информации и метод её тематического анализа на основе частотно-контекстной классификации: дисс. ... канд. тех. наук: 05.13.01/ В.Л. Чугреев. – СПб.: СПГЭУ, 2003. –185 с.

4. Крутояров Д.В. О выборе метода анализа текстовой информации / Д.В. Крутояров // Проблемы полиграфии и издательского дела. – 2006. – № 3. – С. 175-178.

5. Онтологии и тезаурусы: модели, инструменты, приложения курс лекций [Электронный ресурс] / Б.В. Добров, В.В. Иванов, Н.В. Лукашевич, В.Д. Соловьев. – Режим доступа к документу:– http://www.intuit.ru/department/expert/ ontoth/3/2.html, лекция 3. Заголовок с экрана.

6. Адаманский А. Обзор методов и алгоритмов полнотекстового поиска [Электронный ресурс] / А. Адаманский // Новосибирский государственный университет: тезисы доклада 2006 г. – Режим доступа к документу:– http://callisto.nsu.ru /documentation/searchreview.pdf . Заголовок с экрана.

7. Taher Haveliwala. Efficient computation of pagerank. Technical Report 2009-31. Stanford University, 2009. [Электронный ресурс] / Taher Haveliwala. – Режим доступа к документу: http://dbpubs.stanford.edu/pub /2009-31. Заголовок с экрана.

8. Peter Marendy. A Review of World Wide Web searching techniques, focusing on HITS and related algorithms that utilise the link topology of the World Wide Web to provide the basis for a structure based search technology, 2001. [Электронный ресурс] / Peter Marendy. – Режим доступа к документу: http://citeseer .nj.nec. com/559198.html. Заголовок с экрана.

9. String Search Graham A. Stephen October 1999 Technical Report TR-92-gas-01 School of Electronic Engineering Science University College of North Wales Dean Street, Bangor, Gwynedd, UK LL57 1UT.

10. Graham A. Stephen Анализ строк. [Электронный ресурс] / A. Graham. – Режим доступа к документу: http://www.hardline.ru/1/11/1352/1750-1.html. Заголовок с экрана.

11. Смит Б. Методы и алгоритмы вычислений на строках. Теоретические основы регулярных вычислений./ Билл Смит. – М.: Вильямс, 2006. – 496 с.

12. Baeza-Yates R. Handbook of algorithms and data structures (2ed., AW, 1991)(T)(433s)_CsAl_.djvu [Электронный ресурс] / R. Baeza-Yates. – Режим доступа к документу: http://statik.qip-file-serv12.ru/uploads/newlinks/e0/nt10/vnormal/bnone/d43385312/tf4169/s0/G._H._Gonnet_R._Baeza-Yates_-_Handbook_of_Algorithms.rar.

13 Kurtz St. Fundamental Algorithms For A Declarative Pattern Matching System / St. Kurtz. – Bielefeld: Universität Bielefeld, 1995. – 238 c.

14. Lecro T. Exact string matching algorithms. – [Электронный ресурс] / T. Lecro. – Режим доступа к документу: http://algolist.manual.ru/. Заголовок с экрана.

15. Кормен Т. Алгоритмы построения и анализ/ Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО, 2002. – 960 с.

16. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; под ред. И.В. Красикова. – М.: Вильямс, 2005. – 1296 с. – ISBN 5-8459-0857-4.

17. Donald Knuth Vaughan Pratt. Fast pattern matching in strings /Donald Knuth; James H. Morris // Jr . SIAM Journal on Computing 6 (2). – 1977. – № 6 (2). – P. 323-350. DOI:10.1137/0206024.

Поступила в редакцию 28.05.2011

Рецензент: д-р техн. наук, проф. В. М. Вартанян, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДІВ ВИЯВЛЕННЯ ПЛАГІАТУ У РОБОТАХ СТУДЕНТІВ ТЕХНІЧНИХ СПЕЦІАЛЬНОСТЕЙ

І.В. Шостак, І.В. Груздо

У статті наведені результати порівняльного аналізу методів виявлення фактів плагіату в природномовних текстах в аспекті застосування цих методів для виявлення текстологічних запозичень в роботах студентів технічних ВНЗ. На основі результатів проведеного аналізу показано необхідність удосконалення математичного і методичного забезпечення систем пошуку плагіату, спрямованого на підвищення ефективності таких систем. Показано, що в складі математичного забезпечення систем пошуку плагіату в студентських роботах доцільно використовувати алгоритм Бойера - Мура, при цьому виправданим з практичної точки зору виявляється застосування методу зіставлення рядків.

Ключові слова: плагіат, студентський плагіат, природно-язикові тексти, інформаційно-пошукові системи

ANALYSIS OF THE EFFECTIVENESS OF METHODS OF DETECTION OF PLAGIARISM IN THE WORKS TECHNICAL STUDENTS

I.V. Shostak, I.V. Gruzdo

The results of a comparative analysis of methods of detecting evidence of plagiarism in natural language texts in the aspect of applying these techniques to identify textual borrowing in the works of students of technical universities. Based on the results of the analysis shows the need to improve mathematical and methodological support of search engines plagiarism aimed at improving the efficiency of such systems. It is shown that the composition of software systems, finding plagiarism in student papers should be used algorithm Boyer - Moore, with justifiable from a practical point of view is the application of the method of matching rows.

Key words: plagiarism, student plagiarism, natural-language texts, information retrieval systems/

Шостак Игорь Владимирович – д-р техн. наук, проф. кафедры инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: iv_shostak@rambler.ru.

Груздо Ирина Владимировна – аспирантка кафедры инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: tigralwovna@rambler.ru.