УДК 004.42

Ю. Б. ПОПОВА, А. В. ГОЛОБУРДА

АЛГОРИТМИЧЕСКАЯ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОПРЕДЕЛЕНИЯ ПЛАГИАТА В СИСТЕМАХ УПРАВЛЕНИЯ ОБУЧЕНИЕМ

Белорусский национальный технический университет

Основное преимущество применения информационных технологий в образовании, заключающееся в ускорении и упрощении обмена информацией, одновременно является его недостатком, поскольку порождает проблему плагиата. Целью данной работы является разработка программного обеспечения для проверки текста на уникальность в системах управления обучением. Для достижения указанной цели необходимо решить круг задач, связанных с выбором метода определения плагиата, его алгоритмизацией и программной реализацией. В работе рассматриваются методы шинглов, супершинглов, сигнатурные методы, векторные модели представления текста, а также кластерный анализ текстовой информации. Авторами предлагается модификация векторной модели для повышения точности определения схожих документов за счет формирования N-списка каждого документа отдельно. Вследствие этого будет происходить попарное сравнение документов и формирование образа одного документа относительно N-списка другого. Таким образом, в i-й строку матрицы схожести будут записываться коэффициенты схожести всех рассматриваемых документов относительно i-го документа. Предлагаемая модификация также позволит ускорить процесс вычислений, поскольку отсутствует необходимость искать общие термы для всех документов. Для анализа большого количества работ обучающихся с целью проверки их на плагиат авторы предлагают использовать кластерный подход. Его применение показало, что время определения дубликатов для одного документа и для всех документов, входящих в выборку, одинаковое. Т. е. за один и тот же промежуток времени можно получить все варианты одинаковых работ обучающихся. Таким образом, применение кластерного анализа текстовой информации при определении плагиата ощутимо экономит как время преподавателя, так и вычислительные ресурсы. Программная реализация предлагаемых алгоритмов выполнена в виде веб-сервиса на языке Java.

Ключевые слова: *плагиат, векторная модель документа, термы, N-список, матрица схожести, кластер, кластерный анализ.*

Введение

Современное развитие информационных технологий и сети Интернет предоставило широким кругам пользователей доступ к большим объемам информации. Так общедоступными стали методические указания, курсы лекций, учебники и т. д. Одновременно появились коллекции рефератов, готовых лабораторных работ, курсовых и дипломных проектов. Все это способствует распространению и повторному использованию их среди обучающихся.

В последнее время наблюдается бурный рост использования в учебном процессе заимствованной информации, которую можно отнести к разряду плагиата. Согласно определения, приведенного в [1], плагиат – это умышленное присвоение авторства чужого произведения науки или искусства, чужих идей или изобретений. Поэтому задача обнаружения недобросовестного использования чужих работ (фактов плагиата) в учебных заведениях приобретает высокую актуальность.

Для проверки документов на плагиат в сети Интернет существует несколько программных продуктов, рассмотренных в [2]. Однако не все обучающиеся размещают свои работы во всемирной сети и часто передают их младшим курсам. Поэтому в учебных заведениях существует проблема плагиата работ с прежних лет. Одним из решений указанной проблемы является использование автоматизированных систем управления обучением (англ., Learning Management System, LMS) [3]. Все обучающиеся закачивают свои лабораторные, контрольные и курсовые работы в такие системы, обра-

зуя архивы прошлых лет, и преподавателю не составляет труда проверить работы на плагиат. Ключевым моментом здесь является возможность LMS делать такие проверки. Поэтому авторами предлагается веб-сервис, который может быть интегрирован в любую LMS, для поиска заимствованных работ и составления отчетов с указанием процентов схожести с аналогами.

1. Описание методов для определения плагиата. Существует несколько подходов для обнаружения плагиата. Наибольшую популярность получил метод шинглов [4], основанный на представлении текстов в виде множества последовательностей фиксированной длины, состоящих из соседних слов. При значительном пересечении таких множеств документы будут похожи друг на друга. Одной из модификаций данного метода стал алгоритм супершинглов [5], идея которого заключается в применении различных хэш-функций к элементам множества шинглов и выборе для каждой из них шингла, минимизирующего ее значение. Из выбранных шинглов формируются группы, которые называются супершинглами. Если мера сходства наборов супершинглов не меньше заданного значения, то такие документы считаются похожими друг на друга. Данная категория методов показывает высокую точность нахождения дубликатов [5]. Главным недостатком метода шинглов является неустойчивость к небольшим изменениям содержания документа (изменения окончаний, перестановки слов (словосочетаний) и т. д.).

Другой группой методов по поиску плагиата являются сигнатурные методы, детальный обзор которых выполнен в [5]. Суть сигнатурных методов заключается в представлении документа с помощью одного числового значения – сигнатуры. Проверка схожести документов сводится к сравнению их сигнатур. Существуют различные способы определения сигнатур документов:

– использование хэш-функции, вычисленной для всего документа. Данный способ позволяет определить только точные дубликаты;

– использование хэш-функции, вычисленной для строки, которую получили из соединенных в алфавитном порядке нескольких слов документа с наибольшим весом, рассчитанным разными способами;

– использование хэш-функции, вычисленной для строки, которую получили из соединенных в алфавитном порядке нескольких наиболее длинных или «тяжелых» (состоящих из слов с наибольшим суммарным весом) предложений документа.

Данные методы показывают приемлемую точность и просты в реализации. Главный недостаток состоит в том, что сигнатурные методы не учитывают связность текста [5].

В задачах интеллектуальной обработки текстов широко используются векторные модели текстовых документов. Каждый документ представляется в виде вектора $D = (D^1, D^2, \ldots, D^N)$, в котором элементы отражают некоторую характеристику документа [6]. В качестве элементов могут использоваться слова, встречающиеся в текстах. Данная характеристика является наиболее распространенной, однако встречаются и другие, например, частота появления различных пар символов или частота появления различных частей речи [7]. Дальнейшее сравнение документов проводится путем оценки сходства между полученными векторами с использованием различных мер сходства, где наиболее часто применяют коэффициент Дайса или косинусную меру [8]. Преимуществом векторной модели текстовых документов является возможность использования сразу нескольких характеристик, высокая точность и быстрота обработки текстовой информации по сравнению с методом шинглов. К недостатку данного метода можно отнести сложность реализации.

2. Модификация векторной модели для определения плагиата. При создании векторной модели текстового документа на первом шаге исходный текст разбивается на одиночные слова. Далее происходит морфологический разбор документа: слова приводятся в нормализованную форму (именительный падеж, единственное число). Затем подсчитывается количество появлений каждого нормализованного слова в анализируемых документах. Слова упорядочиваются по количеству появлений в порядке убывания. Из полученного набора слов удаляются стоп-слова (например, предлоги, местоимения, наречия, цифры и т. д.). После этого происходит формирование образов документов на основе N наиболее часто встречающихся слов, называемых термами. Образ документа

```
public List<SimilarityRow> makeSimilarityRows(List<Doc> docs, int termCount) {
    List<SimilarityRow> rows = new ArrayList<SimilarityRow>();
    if (termCount == 0)
        termCount = TERM_COUNT;
    //получим список термов для каждого документа
    Map<String, List<String>> doc2terms = getDocumentTerm(docs, termCount);
    Map<String, Integer> doc2similarity = new HashMap<>();

    for(int i = 0; i < docs.size(); i++) {
        Doc doc = docs.get(i);
        SimilarityRow row = new SimilarityRow();
        row.setDoc(doc);
        //список термов для данного документа
        List<String> terms = doc2terms.get(row.getDoc().getDocIndex());
        int[] arrX = initionalize(termCount);
        //сравнение с другими документами
        for (int j = 0; j < docs.size(); j++) {
            //ид документа
            Doc doc2 = docs.get(j);
            //набор термов для документа, с которым сравниваем
            List<String> terms2 = doc2terms.get(doc2.getDocIndex());
            //схожесть одинаковых документов можно не считать. всегда = 100
            if (row.getDoc().equals(doc2)) {
                //сам с собой можно не считать. всегда 100
                row.getSimilarity().put(doc2, 100);
                continue;
            }
            //уже считали раньше
            if (doc2similarity.containsKey(getUniqueKey(i, j))) {
                row.getSimilarity().put(doc2, doc2similarity.get(getUniqueKey(i, j)));
                continue;
            }
            int[] arrY = new int[termCount];
            //если i-е слово из N - списка присутствует в документе,
            //то значением i - го элемента образа документа считается 1, в противном случае — 0.
            for (String term : terms)
                arrY[terms.indexOf(term)] = terms2.contains(term) ? 1 : 0;
            if (arrY.length < termCount) {
                for (int k = arrY.length; k < termCount; k++)
                    arrY[k] = 0;
            }
            //считаем коэффициент схожести
            int coeff = getSimilarityCoefficient(arrX, arrY);
            //добавляем в матрицу
            row.getSimilarity().put(doc2, coeff);
            doc2similarity.put(getUniqueKey(i, j), coeff);
        }
        rows.add(row);
    }
    return rows;
}
```

Рис. 1. Программная реализация модификации векторной модели для определения плагиата

представляет собой одномерный массив длиной N (N-список), формирующийся следующим образом: если i-е слово из N-списка присутствует в документе, то значением i-го элемента образа документа считается 1, в противном случае – 0. На основе полученных образов документов составляется матрица схожести. В качестве коэффициента схожести двух документов используется косинусная мера сходства [9]:

$$\cos\left(\bar{X},\bar{Y}\right)=\frac{\sum_{i=1}^{n}X_iY_i}{\sqrt{\sum_{i=1}^{n}X_i^2}\sqrt{\sum_{i=1}^{n}Y_i^2}}, \qquad (1)$$

где X и Y – образы документов.

Для повышения точности определения схожих документов авторами предлагается модифицировать исходный алгоритм, формируя не общий N-список для всех анализируемых документов, а для каждого документа отдельно. Вследствие чего будет происходить попарное сравнение документов и формирование образа одного документа относительно N-списка другого. Таким образом, в i-й строку матрицы схожести будут записываться коэффициенты схожести всех рассматриваемых документов относительно i-го документа. Предлагаемая модификация также позволит ускорить процесс вычислений, поскольку нет необходимости искать общие термы для всех документов. Программная реализация описанного алгоритма приведена на рис. 1 в виде исходного кода на

языке Java. На вход представленной функции подается список документов для анализа (*docs*) и количество слов для *N*-списка (*termCount*). В списке *rows* будет храниться матрица схожести. Далее в строке 6 определяется *N*-список для каждого документа *doc2terms*. В строках 10–14 определяется документ, для которого будет формироваться строка в матрице. Из заранее подготовленных списков термов рассматриваются те, которые соответствуют сравниваемым документам. Образ проверяемого документа определять нет необходимости, т. к. все элементы будут равны 1 (строка 15). В строках 17–47 вычисляются коэффициенты схожести с другими документами выборки. Элементами главной диагонали матрицы схожести всегда будет 1, т. к. на пересечении *i*-й строки и *i*-го столбца будут одни и те же документы (строки 23–27 на рис. 1, в которых вместо 1 используется число 100, как результат умножения всех коэффициентов на 100, т. к. целые числа занимают меньше памяти, чем вещественные). Поскольку матрица схожести симметрична относительно главной диагонали, подсчитывать некоторые коэффициенты не нужно, они были вычислены ранее (строки программного кода 29–32 на рис. 1). В строках 36–37 формируется образ документа относительно рассматриваемого *N*-списка.

В процессе работы алгоритма может возникнуть ситуация, когда в проверяемом документе количество слов *N*-списка меньше, чем задано в параметре *termCount.* Поскольку для вычисления коэффициента схожести по формуле (1) образы документов должны быть одной длины, то в указанном выше случае произойдет ошибка вычисления. Избежать данной ситуации поможет реализация в строках 38–41 на рис. 1. Если образ документа меньше заданного количества термов, тогда недостающее количество элементов дополняется нулями. Далее по формуле (1) вычисляется коэффициент схожести и добавляется в строку матрицы.

Указанные выше методы для определения плагиата позволяют сравнивать один интересующий документ со всеми остальными. Однако в ходе учебного процесса преподаватель сталкивается с большим количеством работ студентов, которые необходимо проверять одновременно, например, контрольные или курсовые работы целого потока обучающихся. Поэтому проверять на плагиат каждую работу отдельно – это очень затратный процесс как по времени выполнения, так и по вычислительным ресурсам. Для решения данной проблемы можно использовать кластерный анализ текстовой информации.

3. Кластерный подход для решения задачи. Основная цель кластерного анализа – разбить множество объектов на группы таким образом, чтобы объекты внутри одной группы

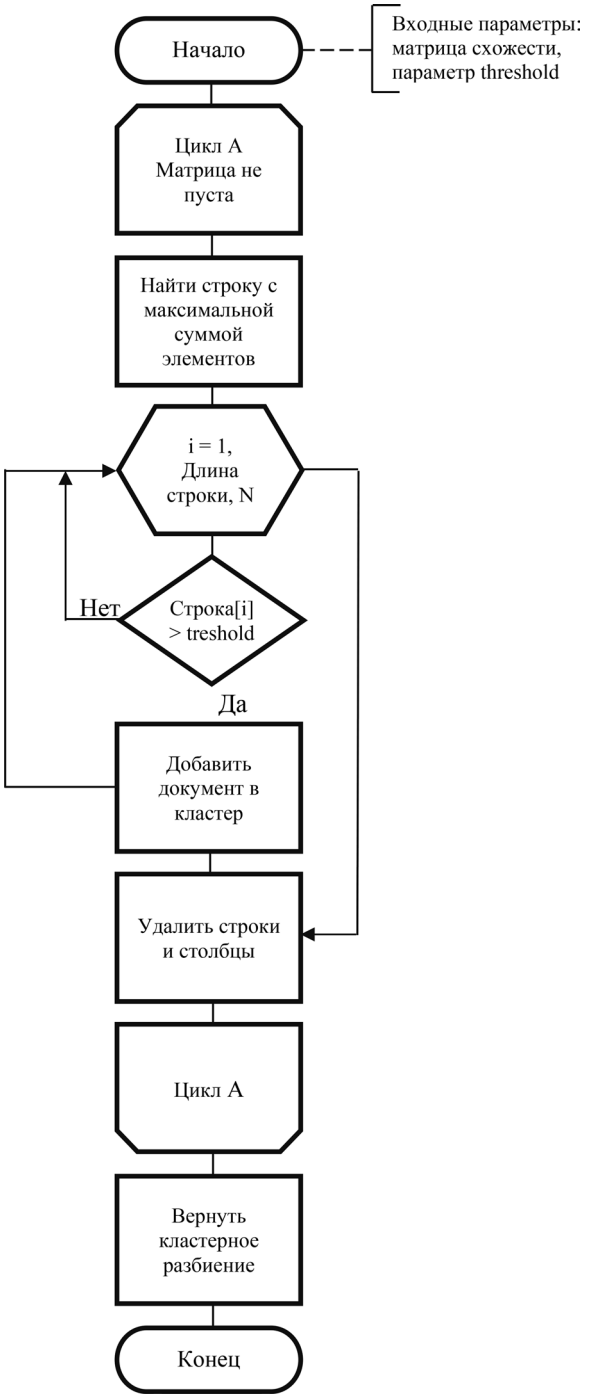


Рис. 2. Схема алгоритма кластерного подхода

```
1  public List<Cluster> clustering(List<SimilarityRow> rows, int threshold) {
2      if (threshold == 0)
3          threshold = THRESHOLD;
4
5      List<Cluster> clusters = new ArrayList<>();
6      int size = rows.size();
7      //пока есть нераспределенные документы
8      while (size > 0) {
9          //ищем строку с максимальной суммой
10         SimilarityRow maxRow = rows.stream().max( (r1, r2) -> r1.getSumSimilarity().compareTo(r2.getSumSimilarity())).get();
11
12         Cluster cluster = new Cluster();
13         for(Map.Entry<Doc, Integer> entry : maxRow.getSimilarity().entrySet()){
14             //если коэффициент схожести больше заданного порогового значения - добавляем документ в кластер
15             if (entry.getValue().intValue() >= threshold)
16                 cluster.getDocs().add(entry.getKey());
17         }
18
19         //кластер не пустой
20         if (!cluster.getDocs().isEmpty() && cluster.getDocs().size() > 1)
21             clusters.add(cluster);
22
23         //удаляем документы, которые попали в кластер
24         deleteRows(rows, cluster.getDocs());
25
26         size = rows.size();
27     }
28
29     return clusters;
30 }
```

Рис. 3. Программная реализация кластерного анализа текстовой информации

были максимально похожими друг на друга, но в то же время максимально отличались от объектов другой группы. В настоящее время существует достаточно большое количество алгоритмов кластеризации, одним из которых является, так называемый, жадный алгоритм, определяющий на каждом шагу локально оптимальный выбор в расчете на приведение к оптимальному решению всей задачи [9].

Рассмотрим модификацию жадного алгоритма, предложенного в [10], и применим ее для решения задачи определения плагиата. На первом шаге в матрице схожести определяется строка с максимальной суммой элементов. Документ, соответствующий данной строке, объявляется центром кластера, а сама строка содержит коэффициенты схожести этого документа со всеми остальными документами набора. На втором шаге в кластер добавляются все документы, коэффициент схожести которых с центром кластера больше или равен некоторого заранее заданного порогового значения. После этого из матрицы удаляются все строки и столбцы, соответствующие добавленным в кластер документам. Шаги 1 и 2 повторяются до тех пор, пока матрица не окажется пуста. Схема алгоритма предлагаемого подхода представлена на рис. 2.

Следует добавить, что кластерный анализ для определения плагиата может работать с матрицей схожести, полученной любым методом из описанных выше. Рассчитанная один раз для определенного набора документов, матрица схожести позволит построить кластеры похожих друг на друга документов.

Алгоритм кластерного анализа текстовой информации реализован на языке программирования Java. Матрица схожести реализована как список строк с коэффициентами схожести. Программный код алгоритма приведен на рис. 3.

На вход представленной на рис. 3 функции, реализующей алгоритм кластеризации, подается матрица схожести *rows* и пороговое значение вхождения документа в кластер *threshold*. В списке *clusters* будут храниться полученные в результате кластеризации группы схожих документов. Для удобства строка матрицы схожести с коэффициентами реализована как отдельный объект *SimilarityRow*, содержащий номер документа, к которому относится строка, и коэффициенты схожести с другими документами. Также для удобства отдельно реализованы методы для подсчета суммы коэффициентов схожести строки и удаления уже попавших в кластер документов. В строке 10 исходного кода ищется строка в матрице схожести с максимальной суммой коэффициентов. В строках 13–17 происходит сравнение максимального коэффициента схожести с параметром *threshold.* Если значение больше заданного порогового, значит этот документ добавляется в кластер. Избежать пустых кластеров или кластеров с одним элементом помогает код в строках 20–21. После добавления документов в кластер происходит удаление из матрицы соответствующих им строк и столб-

цов. В строке 26 уменьшается параметр *size*, хранящий размер матрицы схожести, во избежание зацикливания.

Для проведения вычислительных экспериментов было подготовлено два тестовых набора лабораторных работ объемом 1500–2000 слов по дисциплине «Надежность программного обеспечения». Первый набор состоял из 50 документов, а второй – из 100. Результаты вычислений показали, что время проверки одного документа на плагиат из набора пятидесяти лабораторных работ составляет 88 секунд. Такое же количество времени требуется для проверки на плагиат всех пятидесяти документов, применяя кластерный анализ. С увеличением количества проверяемых документов в два раза время вычислений увеличивается приблизительно в такое же количество раз. Таким образом, проведенные эксперименты показали преимущество использования кластерного анализа при определении плагиата: данный подход позволяет за один раз найти сразу все варианты схожих работ, что существенно экономит время поиска.

В ходе экспериментов было также замечено, что при повторных проверках документов время вычислений может быть существенно сокращено за счет первоначального сохранения результатов морфологического разбора. Поскольку сравнение каждого нового документа происходит с ранее проверенными документами, то для них нет необходимости делать морфологический разбор каждый раз, когда формируется матрица схожести, а можно брать сохраненные варианты предыдущих разборов. Результаты вычислительных экспериментов приведены в таблице.

Результаты вычислительных экспериментов

Количество документов	Среднее время определения плагиата, с	
	Для одного документа	Для группы документов с применением кластерного анализа
Без сохранения результатов морфологического разбора документов		
50	88	88
100	174	174
С сохранением результатов морфологического разбора документов		
50	2	2
100	3	3

Заключение

Применение предложенного алгоритмического и программного обеспечения в системах управления обучением позволит значительно сократить время преподавателя при поиске заимствованных работ обучающихся. Предложенная авторами модификация векторной модели текстового документа позволит повысить точность определения схожих документов и ускорить процесс вычислений за счет их попарного сравнения и формирования образа одного документа относительно *N*-списка другого документа. Применение кластерного анализа текстовой информации после построения матрицы схожести показало, что время определения дубликатов для одного документа и время определения всех схожих документов выборки одинаковое. Таким образом, за одно и тоже время можно получить все варианты заимствованных работ. Сохранение результатов морфологического разбора документов позволит в десятки раз сократить время повторной проверки на плагиат.

Литература

1. **Бобкова, О. В.** Плагиат как гражданское правонарушение / Бобкова О. В., Давыдов С. А., Ковалева И. А. // Патенты и лицензии. – 2016. – № 7. – С. 31–41.

2. **Голобурда, А. В.** Проверка плагиата в веб-приложениях / А. В. Голобурда, Ю. Б. Попова // Информационные технологии в образовании, науке и производстве: IV Международная научно-техническая интернет-конференция, 18–19 ноября 2016 г. Секция Информационные технологии в производстве и научных исследованиях [Электронный ресурс]. – Режим доступа: http://rep.bntu.by/handle/data/27126. – Дата доступа: 25.11.2017.

3. **Попова, Ю. Б.** Классификация автоматизированных систем управления обучением / Попова Ю. Б. // Системный анализ и прикладная информатика. – 2016. – № 2. – С. 51–58.

4. **Broder, A.** On the resemblance and containment of documents / Broder A. // Compression and Complexity of Sequences (SEQUENCES'97). – IEEE Computer Society, 1998. – P. 21–29.

5. **Зеленкова, Ю. Г.** Сравнительный анализ методов определения нечетких дубликатов для Web-документов / Зеленкова Ю. Г., Сегалович И. В. // Труды 9-ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции»: сб. работ участников конкурса. – Переславль-Залесский, 2007. – Т. 1. – С. 169–172.

6. **Моченов, С. В.** Векторная модель представления текстовой информации / С. В. Моченов, А. М. Бледнов, Ю. А. Луговских // Материалы международной научной конференции. – Ижевск, 2006. – С. 133–139.

7. **Андреев, А. М.** Метод обнаружения дубликатов в потоке текстовых документов / Андреев А. М., Березкин Д. В., Козлов И. А., Симаков К. В. // Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». – Дубна, 2014. – С. 310–321.

8. **Антонова, А. Ю.** Об использовании мер сходства при анализе документации / Антонова А. Ю., Клышинский Э. С. // Труды 13-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». – Воронеж, 2011. – С. 134–138.

9. **Баранов, М. А.** Модификация жадного алгоритма кластеризации / Баранов М. А. // Прикладная информатика. – 2013. – № 2. – С. 78–88.

10. **Барахнин, В. Б.** Кластеризация текстовых документов на основе составных ключевых термов / Барахнин В. Б., Ткачев Д. А. // Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2010. – № 2. – С. 5–14.

References

1. **Bobkova, O. V.** Plagiat kak grazhdanskoe pravonarushenie / Bobkova O. V., Davydov S. A., Kovaleva I. A. // Patenty i licenzii. – 2016. – № 7. – S. 31–41.

2. **Goloburda, A. V.** Proverka plagiata v veb-prilozhenijah / A. V. Goloburda, Ju. B. Popova // Informacionnye tehnologii v obrazovanii, nauke i proizvodstve: IV Mezhdunarodnaja nauchno-tehnicheskaja internet-konferencija, 18–19 nojabrja 2016 g. Sekcija Informacionnye tehnologii v proizvodstve i nauchnyh issledovanijah [Jelektronnyj resurs]. – Rezhim dostupa: http://rep.bntu.by/handle/data/27126. – Data dostupa: 25.11.2017.

3. **Popova, Ju. B.** Klassifikacija avtomatizirovannyh sistem upravlenija obucheniem / Popova Ju. B. // Sistemnyj analiz i prikladnaja informatika. – 2016. – № 2. – S. 51–58.

4. **Broder, A.** On the resemblance and containment of documents / Broder A. // Compression and Complexity of Sequences (SEQUENCES'97). – IEEE Computer Society, 1998. – P. 21–29.

5. **Zelenkova, Ju. G.** Sravnitel'nyj analiz metodov opredelenija nechetkih dublikatov dlja Web-dokumentov / Zelenkova Ju. G., Segalovich I. V. // Trudy 9-oj Vserossijskoj nauchnoj konferencii «Jelektronnye biblioteki: perspektivnye metody i tehnologii, jelektronnye kollekcii»: sb. rabot uchastnikov konkursa. – Pereslavl'-Zalesskij, 2007. – T. 1. – S. 169–172.

6. **Mochenov, S. V.** Vektornaja model' predstavlenija tekstovoj informacii / S. V. Mochenov, A. M. Blednov, Ju. A. Lugovskih // Materialy mezhdunarodnoj nauchnoj konferencii. – Izhevsk, 2006. – S. 133–139.

7. **Andreev, A. M.** Metod obnaruzhenija dublikatov v potoke tekstovyh dokumentov / Andreev A. M., Berezkin D. V., Kozlov I. A., Simakov K. V. // Trudy 16-j Vserossijskoj nauchnoj konferencii «Jelektronnye biblioteki: perspektivnye metody i tehnologii, jelektronnye kollekcii». – Dubna, 2014. – S. 310–321.

8. **Antonova, A. Ju.** Ob ispol'zovanii mer shodstva pri analize dokumentacii / Antonova A. Ju., Klyshinskij Je. S. // Trudy 13-j Vserossijskoj nauchnoj konferencii «Jelektronnye biblioteki: perspektivnye metody i tehnologii, jelektronnye kollekcii». – Voronezh, 2011. – S. 134–138.

9. **Baranov, M. A.** Modifikacija zhadnogo algoritma klasterizacii / Baranov M. A. // Prikladnaja informatika. – 2013. – № 2. – S. 78–88.

10. **Barahnin, V. B.** Klasterizacija tekstovyh dokumentov na osnove sostavnyh kljuchevyh termov / Barahnin V. B., Tkachev D. A. // Vestnik Novosibirskogo gosudarstvennogo universiteta. Serija: Informacionnye tehnologii. – 2010. – № 2. – S. 5–14.

Поступила *16.01.2018* | *После доработки* *13.02.2018* | *Принята к печати* *15.03.2018*

Y. B. Popova, A. V. Goloburda

ALGORITHMIC AND PROGRAM IMPLEMENTATION OF THE PLAGIARISM DEFINITION IN LEARNING MANAGEMENT SYSTEMS

Belarusian National Technical University

The main advantage of using information technologies in education, which consists in speeding up and simplifying of information exchange, is also its drawback, because it raises the problem of plagiarism. The purpose of this paper is to develop testing text software for uniqueness in learning management systems. To achieve this goal, it is necessary to solve a range of problems related to the choice of a method for determining plagiarism, its algorithmization and software implementation. The work deals with the methods of shingles, super-shingles, signature methods, vector models of text representation, as well as cluster analysis of text information. The authors suggest a modification of the vector model to improve the accuracy of determining similar documents by creating an N-list of each document separately. As a result, a pairwise comparison of the documents and the formation of the image of one document relative to the N-list of the other will occur. Thus, in the i-th row of the similarity matrix, the coefficients of similarity of all the documents considered relative to the i-th document will be recorded. The proposed modification will also speed up the calculation process, since there is no need to search for common terms for all documents. To analyze a large number of student's works in order to test them for plagiarism, the authors propose using a cluster approach. Its application showed that the time for determining duplicates for one document and for all documents

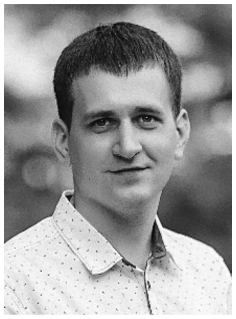
included in the sample is the same. For the same time it is possible to get all the options for the same works of students. Thus, the use of cluster analysis of text information in determining plagiarism significantly saves both the teacher's time and computing resources. The software implementation of the proposed algorithms is implemented as a web service in the Java language.

Keywords: *plagiarism, vector document model, terms, N-list, similarity matrix, cluster, cluster analysis.*

Yuliya B. Popova, PhD, Associate Professor of the Software Department of the Belarusian National Technical University. Her research interests focus on methods and algoritms of optimization in technical systems, engineering of adaptive learning systems and learning management systems (LMS), modeling of student knowledge, software testing and quality assurance.

Email: julia_popova@mail.ru

Попова Юлия Борисовна, доцент, кандидат технических наук, доцент кафедры программного обеспечения вычислительной техники и автоматизированных систем БНТУ. Ее научные интересы связаны с методами и алгоритмами оптимизации технических систем, разработкой адаптивных обучающих систем, автоматизированных систем управления учебным процессом, моделированием знаний, а также с вопросами тестирования и качества программного обеспечения.



Alexander Goloburda received the graduate degree in software engineering from the Belarusian National Technical University in 2016 and the Master's degree in system analysis and control of information processing in 2015. He is currently working on Master's degree program in system analysis and control of information processing. His research interests are related to the analysis of text information, the clustering of text information and the development of web applications.

Email: goloburda.av@gmail.com

Голобурда Александр Вячеславович закончил Белорусский национальный технический университет по специальности «Программное обеспечение информационных технологий» в 2016 году. В настоящее время является магистрантом кафедры программного обеспечения вычислительной техники и автоматизированных систем БНТУ. Его научные интересы связаны с анализом текстовой информации, кластеризацией текстовой информации, а также с разработкой веб-приложений.