Московский Государственный Университет

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра системного программирования

Дипломная работа

Исследование и разработка методов автоматического поиска заимствований в русскоязычных текстах

Выполнил:

Студент 527 группы

Бабаков Александр Валентинович

Научные руководители:

Недумов Ярослав Ростиславович

ак. РАН, проф. Иванников Виктор Петрович

Москва

2014

Оглавление

Аннотация 3

Введение 4

1 Постановка задачи 7

2 Обзор существующих решений 8

2.1 Метод «Шинглов» 8

2.2 Метод супершинглов (мегашинглов) 8

2.3. Surrounding Context N-Grams 9

2.4 I-Match 10

2.5 Метод расчета коэффициента совпадения документов 10

2.6 Метод «Опорных слов» 11

2.7 Методы скрытия факта заимствования в тексте 11

2.8 Выводы 12

3 Исследование и построение решения задачи 14

3.1. Алгоритм по борьбе с модификациями при заимствованиях 14

3.2 Алгоритм для поиска заимствований 18

3.3 Получение набора текстов для поиска 20

3.4 Экспериментальная проверка 21

3.4.1 Экспериментальный тестовый набор на основе "Диссернет" 21

3.4.2 Экспериментальный тестовый набор на основе новостей 24

4 Описание практической части 27

4.1 Общая схема работы 28

4.2 Общая архитектура системы 30

4.3. Характеристики функционирования 31

Заключение 32

Список литературы 33

Приложение А. 35

Аннотация

Работа посвящена исследованию и разработке методов поиска заимствований. В дипломной работе были рассмотрены существующие методы скрытия факта заимствования, а также методы борьбы с ними, рассмотрены различные современные методы поиска заимствований в текстах. В ходе работы был разработан метод, позволяющий производить анализ текста на наличие заимствований из определенной заранее коллекции документов, причем метод обладает определенной устойчивостью к модификациям. Также была разработана автоматизированная система с графическим интерфейсом, реализующая описанный метод для русского языка, было создано две русскоязычных тестовых выборки с различными характеристиками и проведено тестирование.

Введение

Текстовые заимствования являются одной из самых старых форм заимствований, при этом проблема обнаружения текстовых заимствований до сих пор остаётся сложной для решения [3]. С этой проблемой сталкиваются, например, в следующих задачах: установление нарушения авторских прав при незаконном копировании[7], поиск избыточной информации в библиотеках документов, обнаружение спама [2], поиск плагиата.

Текстовый плагиат является частным случаем заимствований в текстах. Плагиат - это выдача чужого произведения за своё или использование в своих трудах чужого произведения без ссылки на автора [3]. Поэтому, заимствование текста с указанием ссылки на автора не является плагиатом. Задача определения корректности заимствования в этой работе не рассматривается.

При заимствованиях может использоваться как часть текста, так и весь текст целиком. Заимствования могут производиться с точным копированием или копированием с модификацией. Во втором случае текст может быть переформулирован (перестановка слов, предложений, замена слов на синонимы), переведен на другой язык, часть информации может быть удалена. Кроме того, если в случае обычных поисковых систем авторы текстов заинтересованы в том, чтобы алгоритмы поиска зафиксировали соответствие текста запросу, то в случае плагиата авторы всевозможно препятствуют этому. Например, для скрытия факта плагиата применяется изменение формата текста (добавление пробелов, изменение пунктуации, создание “невидимых” символов, замена символов кириллицы символами латинского алфавита и т.д.).

Развитие Интернет приводит к тому, что информация, попадая в открытый доступ, становится доступной всем желающим, что приводит к учащению случаев заимствования [11]. Со временем число документов постоянно увеличивается (и, следовательно, увеличивается число потенциальных источников плагиата): в случае поиска источников в Интернет, по данным Google, на декабрь 2012 года в Интернете существует 634 миллиона сайтов [5], на которых нужно производить поиск. Такой объем данных делает затруднительным подробный попарный анализ текстов, т.к. это занимает большое время: сравнение всевозможных подпоследовательностей текстов имеет сложность $O(n^2)$ (n – это длина документа) [12], а с учетом того, что плагиат может производиться с модификацией, сложность еще более возрастет (т.к. сверх этого необходимо перебрать возможные модификации). Так как ручная проверка текста на наличие заимствований занимает продолжительное время, справедливо утверждение, что ручная проверка текста

на заимствования становится крайне сложной. Для уменьшения трудоёмкости поиска заимствований возможен поиск не во всём Интернет, а в некоторой локальной коллекции документов.

Также дополнительной сложностью анализа текста на заимствования является то, что возможные модификации текста при заимствованиях зависят от используемого языка, и при анализе текста необходимо учитывать специфику этого языка. Зависимости проявляются в различии правил построения предложений, а также возможностях языка по передаче такого же смысла другими словами.

Для решения задачи поиска заимствований создаются автоматические средства поиска заимствований. С 2009 года проводится ежегодный конкурс автоматических средств поиска плагиата [3]. Специально для этого конкурса создана система для построения размеченных корпусов текста с плагиатом, а также тестовая выборка. Выборка состоит только из текстов на английском языке. За несколько лет проведения этого конкурса было выявлено, что большинство алгоритмов поиска заимствований (плагиата) имеет следующую структуру [3]:

1. Выбор кандидатов. Из корпуса отбираются документы, для которых вероятность того, что они являются источниками плагиата, достаточно высока. Важной характеристикой этого этапа является скорость работы, т.к. этот этап должен уменьшить большое число кандидатов до некоторого приемлемого количества, чтобы общее время работы алгоритма оставалось относительно небольшим. В случае небольшой выборки данных (например, поиск источника плагиата среди нескольких сотен документов) этот этап может быть опущен.

2. Подробный анализ. Для каждого из документов-кандидатов анализируется степень сходства с проверяемым текстом. При выполнении некоторых условий (зависящих от конкретного алгоритма) предполагается случай заимствования. Если этот шаг выполнять без этапа выборки кандидатов, то анализ большого числа документов займет существенное время.

3. Пост-обработка. Полученные на втором этапе результаты проверяются человеком. Это необходимо, например, чтобы отделить ложные срабатывания в случае корректных заимствований (например, цитирование) от плагиата, или же для обеспечения точности результата. В случае, если это не является необходимым, этот этап также может быть опущен.

База документов, используемая при поиске заимствований, также влияет на алгоритмы. В случае указанного выше конкурса для доступа к базе документов используется поисковая система ChatNoir, и для получения документов для анализа на плагиат необходимо произвести поисковый запрос через эту систему. Тем самым решается первый этап поиска заимствований, описанный выше: выбор кандидатов.

Поскольку конкурс проводится для англоязычных текстов, применимость построенных на конкурсе алгоритмов не проверялась для русского языка, так же как и отсутствуют размеченные тестовые наборы для русского языка. Существующие автоматизированные системы поиска плагиата, способные производить поиск плагиата на русском языке, скрывают устройство работы из-за коммерческой тайны.

Целью данной работы является разработка системы поиска заимствований в корпусах русскоязычных текстах, причем поиск должен производиться в локальной базе. Поиск заимствований должен производиться с учетом возможных модификаций при заимствовании, и система должна быть способна выделять конкретные фрагменты заимствованного текста, а также соответствующие им фрагменты текста-источника.

1 Постановка задачи

Для достижения поставленной цели необходимо решить следующие задачи:

1. Рассмотреть существующие алгоритмы определения заимствований в текстах.

2. Рассмотреть существующие методы скрытия факта заимствования в текстах, а также методов борьбы с ними.

3. Разработать метод поиска заимствований в русскоязычных текстах, устойчивый к модификациям при заимствовании.

4. Создать программное средство на основе разработанного метода, обеспечивающее поиск заимствований в текстах с возможностью визуализации заимствованных фрагментов текста в проверяемом документе и документе-источнике. Поиск заимствований должен осуществляться в локальной базе документов. Реализация должна быть выполнена на языке Java.

5. Провести оценку качества разработанного программного средства в терминах точности, полноты, F-меры и производительности. Для оценки качества необходимо построить русскоязычный тестовый набор.

2 Обзор существующих решений

Для поиска заимствований применяются различные методы, которые могут применяться на различных этапах поиска заимствований: отборе документов, или же определении конкретного заимствования.

Рассмотрим существующие методы поиска заимствований. Основными критериями сравнения будут являться применимость для обнаружения конкретных заимствований, т.е. выделения границ заимствованного фрагмента текста, применимость для отбора документов, устойчивость к модификациям текста и качество алгоритма (точность, полнота). Также рассмотрим возможные методы скрытия факта заимствования.

2.1 Метод «Шинглов»

В 1997 году A. Broder[9] предложил метод, основанный на представлении документа в виде последовательностей фиксированной длины N, состоящих из соседних слов. При этом на последовательности могут накладываться ограничения, например, слова должны находиться в одном предложении. Такие последовательности в одних источниках называют "шинглами" [1], в других "N-граммами" [4]. Два документа считаются похожими, если множества их N-грамм существенно пересекаются. Аналогично можно считать похожесть двух предложений, или же предложения и текста. Число N-грамм для каждого документа является достаточно большим, используются различные способы усечения их множества, например автором был предложен следующий метод для уменьшения числа шинглов: оставлять только те шинглы, для которых статистические функции принимают фиксированные значения.

Алгоритм не может использоваться для определения конкретных заимствований, так как любая модификация текста при заимствовании приведет к тому, что шингл изменится, и случай заимствования обнаружен не будет, что приведет к уменьшению полноты. Алгоритм может использоваться для отбора документов на основе того, считаются ли проверяемый документ и возможный документ-источник похожими (см. выше), но в случае использования этого алгоритма полнота будет низкой.

2.2 Метод супершинглов (мегашинглов)

Модификация алгоритма шинглов от D. Fetterly[10] состоит в том, что каждый документ представлялся 84 шинглами. Выбор из всего множества шинглов происходит по следующей схеме: для всех шинглов документа рассчитывается значение 84 хеш функций. Для каждой хеш функции выбирается шингл с максимальным значением хеш функции. Затем эти 84 шингла разбиваются на 6 групп по 14 шинглов. Такие группы называются

"супершинглами". Далее документ представляется всевозможными попарными сочетаниями из 6 супершинглов, которые называются "мегашинглами". Число таких мегашинглов равно 15 (число сочетаний из 6 по 2). Два документа сходны по содержанию, если у них совпадает хотя бы один мегашингл.

Алгоритм является неустойчивым при модификации текста при заимствованиях, так как при модификации текста, соответствующего шинглу изменится также и мегашингл. Также алгоритм не обнаруживает заимствования в случае малого совпадения документов: согласно статье [1], в случае совпадения документов на 96 процентов 2 супершингла совпадают с вероятностью 49 процентов. Соответственно, заимствование одного абзаца в двух относительно больших статьях будет недостаточным для обнаружения этим алгоритмом.

Описанный алгоритм применим для отбора документов для поиска, если критерием отбора документов является масштабные заимствования, для поиска конкретных заимствованных фрагментов алгоритм не применим, так как мегашингл содержит лишь небольшое подмножество шинглов документа, и, соответственно, не может содержать информацию о всех возможных заимствованиях. Также алгоритм не является применимым из-за неустойчивости к модификациям: любая модификация текста приведет к изменению соответствующих шинглов, что приведет к низкой полноте алгоритма.

2.3. Surrounding Context N-Grams

Алгоритм Surrounding Context N-Grams является модификацией алгоритма шинглов, предложенной Diego A. Rodríguez Torrejón и José Manuel Martín Ramos [8]. В основу алгоритма берется предположение о том, что текст может модифицироваться путём удаления единичных слов, грамматических ошибок, перестановкой слов, и стандартный алгоритм шинглов не обнаружит совпадения. Для этого увеличивается число генерируемых шинглов. Для каждой последовательности слов N берется некоторое число M, меньшее N, и из последовательности слов N убирается M слов всевозможными способами. Оставшиеся слова сортируются в лексикографическом порядке и представляют собой новые N-граммы. Недостатком этого алгоритма является существенный рост числа шинглов, что приводит к существенному замедлению работы программы. Также это приводит к увеличению числа ложных срабатываний.

Описанный алгоритм является устойчивым к небольшим модификациям текста (если модифицируются не более M слов подряд), и применим для поиска конкретных заимствований, однако не является применимым при модификации более M слов подряд.

Задать константу M достаточно большой не является возможным, так как это приведет к серьезному уменьшению точности алгоритма. Также может применяться для предварительного отбора документов по признакам, аналогичным стандартному алгоритму шинглов.

2.4 I-Match

В методе I-Match [14] создаётся словарь L, который включает слова со средними значениями IDF[16], поскольку такие слова обеспечивают, как правило, более точные результаты при обнаружении заимствований. Слова с большими и маленькими значениями IDF, которые представляют собой очень редкие, или же наоборот, часто встречающиеся слова, отбрасываются. Затем для каждого документа формируется множество U различных слов, входящих в него, и определяется пересечение U и словаря L. Если размер этого пересечения больше некоторого минимального порога (определяемого экспериментально), то список слов, входящих в пересечение упорядочивается, и для него вычисляется I-Match сигнатура (хеш-функция SHA1).

Два документа считаются похожими, если у них совпадают I-Match сигнатуры (имеет место коллизия хеш кодов). Алгоритм имеет высокую вычислительную эффективность. Основной недостаток — неустойчивость к небольшим изменениям содержания документа.

Существует модификация этого алгоритма с более низкой вычислительной эффективностью, но увеличенной. После создания словаря L создаётся N-1 различных словарей, получаемых путем случайного удаления порядка 35% слов от исходного объема L. В итоге получают N словарей. Для каждого документа вычисляется N I-Match сигнатур, и два документа считаются похожими, если определенное число сигнатур совпадает.

Алгоритм не применим для определения конкретных заимствований, так как позволяет выделить только похожие документы, поэтому применим для фильтрации документов при их существенном совпадении. При модификации слов, входящих в словари алгоритм не является устойчивым, что приводит к уменьшению точности.

2.5 Метод расчета коэффициента совпадения документов

В методе расчета коэффициента совпадения документов [5] сравнение двух текстов происходит по следующей схеме: берутся слова из обоих документов, нормализуются, удаляются дубликаты. Получается список L нормализованных слов длины m. Каждый из документов представляется вектором x длины m, где значением координаты x_i является число вхождений слова L_i. Далее считается коэффициент k

похожести документов как косинус между векторами. Значение k сравнивается с порогом d: если полученный коэффициент больше, то фиксируется случай заимствования. Порог d является параметром алгоритма. По аналогичной схеме также возможно сравнивать также части документов, например предложения.

Алгоритм также не применим для определения конкретных заимствований, но позволяет выделить похожие документы в случае их существенного совпадения. При модификации слов также не является устойчивым, что приводит к уменьшению полноты.

2.6 Метод «Опорных слов»

Метод "Опорных слов" [6] является модификацией предыдущего метода. Сначала из множества документов по выбирается множество из N слов (N — определяется экспериментально), называемых "опорными". Затем каждый документ представляется N-мерным двоичным вектором, где i-я координата равна 1, если i-е "опорное" слово имеет в документе относительную частоту выше определенного порога (устанавливаемого отдельно для каждого "опорного" слова), и равна 0 в противном случае. Этот двоичный вектор называется сигнатурой документа. Два документа «похожи», если у них совпадают сигнатуры (или очень близки, близость считается по расстоянию Хэмминга).

Общие соображения для построения множества "опорных" слов таковы:

1. Множество слов должно охватывать максимально возможное число документов.
2. Число слов в наборе должно быть минимальным.

Алгоритм не применим для определения конкретных заимствований, но позволяет выделить похожие документы в случае их существенного совпадения. При модификации "опорных" слов также не является устойчивым, что приводит к уменьшению полноты.

2.7 Методы скрытия факта заимствования в тексте

Согласно данным исследования [11] модификации часто используются при плагиате, причем в случае осведомленности о факте проверки на плагиат попытки скрыть факт плагиата резко возрастает. Описанные алгоритмы являются неустойчивыми к тем или иным модификациям текста, что делает алгоритмы неэффективными при отсутствии дополнительных средств борьбы с этими модификациями. Для того, чтобы повысить эффективность этих алгоритмов, необходимо использовать методы борьбы со всеми видами модификаций. Для этого рассмотрим каждую из описанных в [11] модификаций по отдельности.

Для скрытия факта заимствования от автоматизированных средств используется следующий метод замены символов. Символы в тексте заменяются другими, похожими или неотличимыми по написанию, но имеющие различные числовые значения в кодах. Например, замена заглавной русской ‘Н’ (эн) большой английской ‘H’ (аш). Это приводит к тому, что автоматизированные средства поиска заимствований считают эти слова разными, и факт заимствования остается необнаруженным. Также могут заменяться и знаки препинания: точки на запятые, и наоборот.

Для усложнения работы автоматических анализаторов плагиата может также использоваться добавление в текст символов, не несущих какой либо информации, например, большого числа подчеркиваний в качестве разделителя абзацев, или обрамлений с помощью символов псевдографики.

Другим часто используемым подходом является корректировка родов, чисел и времён входящих в текст слов. Например, замена слова «сделал» на «сделала» или «сделали», использование местоимения «я» вместо «мы» в оригинальном тексте и т.д.

Еще одним часто используемым подходом скрытия факта плагиата является синонимизация текста, т.е. замена некоторых слов на синонимы, например слова «большой» на «огромный», и в результате текст имеет различное написание. Существуют также автоматизированные средства, позволяющие скрывать факт плагиата синонимизацией. Эти автоматизированные средства производят в тексте замену слов по словарям.

Другим подходом скрытия факта плагиата является перестановка слов без искажения смысла, например, переставить местами прилагательные, относящиеся к одному существительному.

Также средства форматирования текста позволяют преобразовать текст так, что визуально он будет неотличим от оригинала, однако на самом деле текст будет разный. Для этого можно использовать цвет букв неотличимый от цвета фона.

Текст может быть также изменен путем добавления или удаления незначимых слов. Такие слова называют стоп словами. Они не несут самостоятельной смысловой нагрузки, поэтому их можно удалить без искажения смысла текста.

2.8 Выводы

Ни один из описанных алгоритмов поиска заимствований не удовлетворяет полностью одновременно всем поставленным требованиям, т.е. не является устойчивым к

произвольным модификациям текста, и позволяет производить предварительную фильтрацию документов с последующим определением конкретных заимствованных фрагментов.

Наиболее полно удовлетворяет поставленным требованиям алгоритм Surrounding Context N-Grams, который позволяет производить поиск конкретных заимствований, применим для фильтрации, и устойчив к некоторым видам модификаций, однако этот алгоритм не проверялся для использования с описанными выше методами модификаций при заимствования и не применялся для русского языка. Также на этот алгоритм показал высокие результаты тестов ежегодного конкурса систем поиска плагиата [3, 8].

3 Исследование и построение решения задачи

Как было показано в предыдущем разделе, в настоящее время не существует метода поиска заимствований, полностью удовлетворяющего требованиям поставленной задачи. Поэтому было решено модифицировать наиболее эффективный алгоритм Surrounding Context N-Grams и создать на его основе автоматизированную систему поиска заимствований. Этот алгоритм было решено использовать на обоих этапах поиска (как предварительной фильтрации, так и поиска конкретных заимствований), так как описанные выше другие методы предварительной фильтрации не позволяют производить обнаружить заимствование при небольших объёмах заимствования относительно всего размера текста.

Для модификации алгоритма и проверки модификаций на эффективность необходимо решить следующие задачи:

1. Разработать конкретные методы борьбы с модификациями в тексте.
2. Разработать алгоритм для поиска заимствований в тексте с учетом методов борьбы с модификациями, и разработать на основе описанного алгоритма автоматическую систему.
3. Построить тестовые наборы для анализа построенной автоматической системы поиска заимствований.
4. Провести тестирование разработанного алгоритма на полученных тестовых наборах.

Ниже подробно рассмотрена каждая из этих задач.

3.1. Алгоритм по борьбе с модификациями при заимствованиях

Для борьбы с возможными модификациями при заимствованиях было решено производить нормализацию текста, при которой текст будет приводиться к нормальной форме, в которой возможные модификации текста при заимствованиях будут изменяться таким образом, что исходный и модифицированный текст будут иметь одинаковую нормальную форму. Нормализация проводится в соответствии с каждой описанной выше возможной модификацией текста:

1. **Нормализация алфавита**. Так как разрабатываемая система должна работать с русскоязычными текстами, то в текстах предполагается кириллический алфавит. Поэтому для всех символов в тексте происходит замена на кириллические аналоги. Для замены используется словарь, в котором обозначены все пары символов для замены. Пары

одинаковых символов были составлены для английского и кириллического алфавита. Пример работы этого шага изображен на рисунке 1.

Ниагарский водопад	=>	**Ниагарский** водопад
Символ Н - латинская "АШ"		Символ Н - русская "ЭН"

Рис. 1 Пример нормализации алфавита

2. **Нормализация формата текста**. Из текста удаляются все лишние разделители, символы, не являющиеся буквами английского или русского алфавита и цифрами. Все допустимые символы определяются специальным словарем допустимых символов. Пример работы этого шага отображен на рисунке 2.

До преобразования

§ История

Первый опубликованный стеммер был написан Джули Бет Ловинс

После преобразования:

История

Первый опубликованный стеммер был написан Джули Бет Ловинс

Рис. 2 Пример нормализации формата текста

3. **Нормализация слов.** Для решения с изменением чисел, родов и падежей необходимо выделить у слов текста основы, т.к. предполагается, что окончания у слов могут быть изменены. Процесс нахождения основы слова для заданного исходного слова называется стеммингом. Для русского языка существует несколько алгоритмов стемминга.

Идея стеммера Портера[17] состоит в том, что в русском языке существует ограниченное количество словообразующих суффиксов, и стемминг слова происходит без использования каких-либо баз с основами слов, используется только набор суффиксов. Алгоритм отсекает от слова известные суффиксы до тех пор, пока получаемое слово удовлетворяет правилам (например, после удаления должна содержаться хотя бы одна гласная).

Алгоритм Stemka[18] разработан Андреем Коваленко в 2002 году. Основан на вероятностной модели: из слов обучающей выборки выделяются последние две буквы

основы и суффикс, если такой элемент присутствует в модели, то увеличивается её вес, иначе добавляется в модель. После этого массив данных ранжируется по убыванию веса, элементы с весами ниже заданного порога удаляются. В качестве результата получается набор потенциальных окончаний с условиями на предшествующие символы. При разборе слово сканируется по построенным таблицам перехода.

Стеммер Mystem[19] разработан Ильей Сегаловичем в 1998 году. На первом шаге при помощи дерева суффиксов определяются возможные границы между основой и суффиксом, после чего для каждой потенциальной основы проверяется её наличие в словаре. Если основа не будет найдено в основе, то результатом будет набор гипотез, иначе одна гипотеза для словарного слова.

Для усечения окончаний было решено использовать стеммер Портера, т.к. он является наиболее вычислительно эффективным среди описанных стеммеров. Пример работы описанного шага отображен на рисунке 3.

До преобразования: красивые облака
После преобразования: красив облак

Рис. 3 Пример нормализации слов

4. Десинонимизация текста. Для решения проблемы замены слов на синонимы используется словарь для приведения слов к канонической форме. Для каждой группы синонимов нужно выбрать какое-то одно слово, и производить замену любого слова из этой группы на это слово при обработке. Для получения базы синонимов была использована электронная версия словаря Абрамова “Словарь русских синонимов и сходных по смыслу выражений”[20]. Из словаря были выделены множества синонимов, и в качестве основного слова для каждого множества было выбрано первое по порядку слово в этом множестве. Т.к. на этапе нормализации слов в тексте у слов были удалены окончания, в словаре также была произведена нормализация текста. Пример работы данного этапа показан на рисунке 4.

До преобразования: симпатичн девушк
После преобразования: красив девушк

Рис. 4 Пример десинонимизации текста

5. Удаление стоп слов. Для решения проблемы стоп слов используется словарь стоп слов. Каждое слово из текста анализируется на совпадение со стоп словами из словаря, при

совпадение слово удаляется из текста. На рисунке 5 изображено удаление стоп слова "Более того".

До преобразования: Более того, Ан-74 первоначально был переделан
После преобразования: Ан-74 первоначально был переделан

Рис. 5 Пример удаления стоп слов

6. **Сортировка слов в предложении**. Для решения проблемы перестановки слов используется сортировка слов в предложении в лексикографическом порядке. Пример работы данного этапа показан на рисунке 5.

До преобразования: Большой адронный коллайдер
После преобразования: Адронный большой коллайдер

Рис. 6 Пример сортировки слов в предложении

В результате использования всех описанных шагов текст претерпевает существенные изменения (см. пример на рисунке 7), что позволяет привести к общему тексту. Данная последовательность нормализации была выбрана исходя из соображений зависимости от входных данных. Нормализацию алфавита необходимо производить в первую очередь, чтобы на следующих этапах не производить проверку на символы, схожие по написанию. Генерацию шинглов нужно производить в последнюю очередь, когда удалены все слова, подлежащие фильтрации на одном из этапов. Некоторые этапы, такие как десинонимизация и удаление стоп слов можно производить в произвольном порядке.

До преобразования: Более того, Ан-74 первоначально был переделан из опытного Ан-72 для эксплуатации в районах Арктики и Антарктиды под обозначением Ан-72А «Арктический»

После преобразования (нормальная форма): ан арктик арктическ антарктид использован обозначен опытн передел район

Рис. 7 Пример использования совокупности шагов алгоритма

Так как для итоговой визуализации результата необходим исходный текст, необходимо хранить исходный, т.е. немодифицированный текст и связи, позволяющие

восстановить соответствие слов исходного и модифицированного текстов. Для каждого слова сохраняется его соответствие с исходным словом в исходном тексте (координаты начала и длина слова).

3.2 Алгоритм для поиска заимствований

Для поиска заимствований в обработанном тесте было решено использовать модифицированный алгоритм Surrounding Context N-Grams. В отличие от стандартного алгоритма шинглов, где генерация шинглов происходит с фиксированной длиной шингла N, происходит также дополнительная генерация шинглов для предложений с количеством слов меньших N. Это сделано для того, чтобы алгоритм мог находить заимствования предложений с количеством слов меньших N.

Соответственно, создание шинглов происходит по следующий схеме (см. рис 8): на вход алгоритму поступает текст (модифицированный согласно пункту 3.1). Текст разбит на предложения. Далее, если предложение имеет длину в словах, меньшую N-M, то это предложение уже является шинглом, и его необходимо запомнить. Иначе происходит выбор первых N слов в предложении, и из этих N слов всевозможными способами удаляется M слов. Полученные в результате слова запоминаются как очередной шингл.

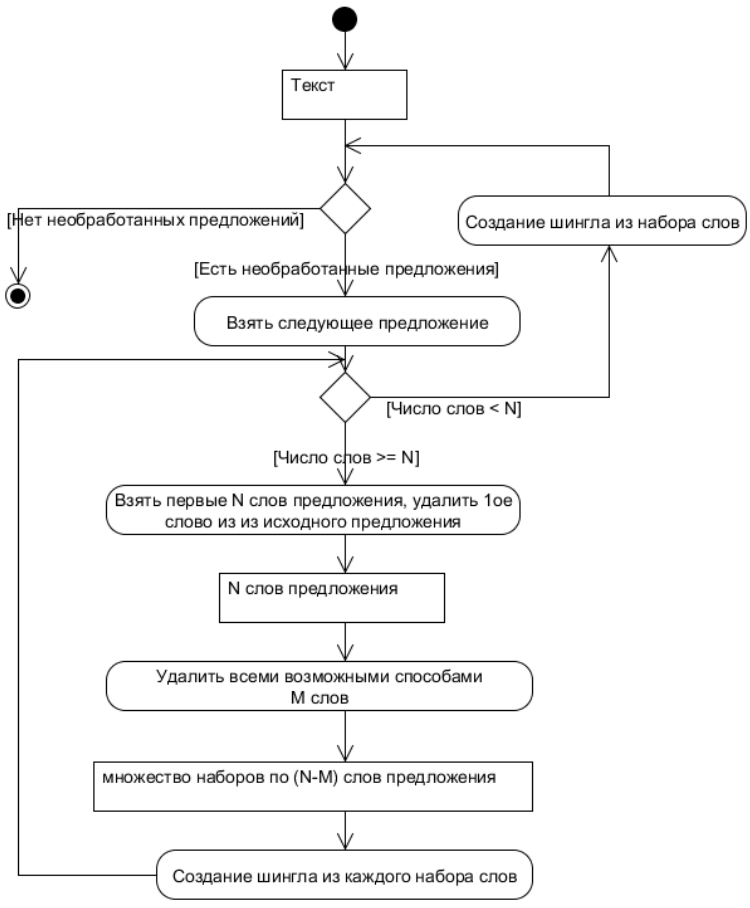

Рис. 8 Схема работы алгоритма

Пример генерации шинглов для N=7 и M=1 показан на рисунке 9. Пример приводится без учета работы предыдущих этапов алгоритма.

Исходное предложение: Более того, Ан-74 первоначально был переделан из опытного Ан-72 для эксплуатации в районах Арктики и Антарктиды под обозначением Ан-72А «Арктический»

Шингл 1: того Ан-74 первоначально был переделан из

Шингл 2: Более Ан-74 первоначально был переделан из

...

Рис. 9 Пример генерации шинглов

Описанный метод генерации шинглов позволяет бороться с удалением до M слов в предложении, а с учетом удаления стоп слов на этапе нормализации текста это число может быть выше в отдельных случаях. Описанный алгоритм может обнаружить плагиат размером не менее N-M слов, при заимствовании в меньших объемах алгоритм этого не обнаружит.

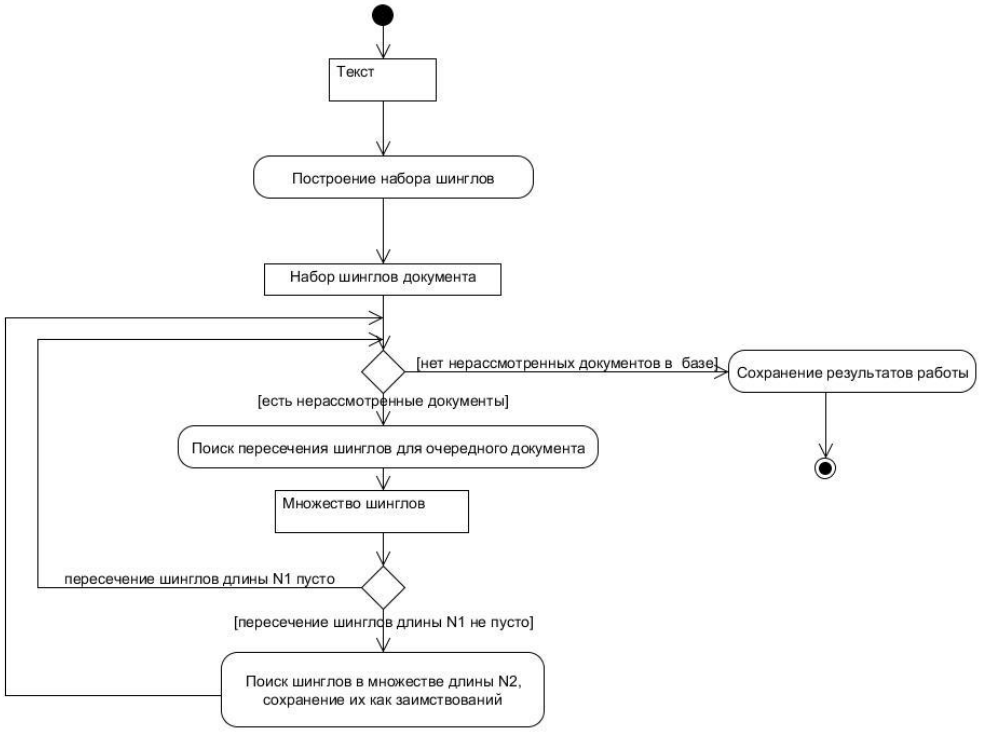

Рис. 10 Схема работы алгоритма поиска заимствований

Непосредственно сам поиск происходит по следующей схеме (см. рисунок 10). Для данного на вход документа по описанному выше алгоритму строится набор шинглов. Для каждого из проверяемых документов также построены шинглы. Поиск проводится в два этапа: выбор документов для поиска и непосредственно сам поиск. Документ участвует во второй этапе в том случае, если в нем имеется хотя бы шингл, совпадающий с шинглом проверяемого документа. На втором этапе происходит попарное сравнение шинглов, при совпадении фиксируется случай заимствования. Результатом работы алгоритма является наборы совпадающих шинглов для каждого из отобранных документов. При этом на этапе выбора документов могут использоваться различные длины шинглов по сравнению с этапом попарного сравнения документов (на рисунке 10 это значения N1 и N2).

3.3 Получение набора текстов для поиска

Поиск документов производится по локальной базе документов. Для оптимизации времени работы для документов локальной базы уже выполнены этапы нормализации и генерации шинглов, что позволяет при проверке документа сразу перейти к этапам фильтрации и поиску конкретных заимствований. Документы в локальной базе имеют специальный формат (см. рисунок 11), который описывает исходный текст, а также некоторые значения множества шинглов документа, необходимые для сравнения и визуализации, а именно позиция начала шингла в документе, конца, длина шингла в словах и значение хеш функции от шингла.

```
<text> Более того, Ан-74 первоначально был переделан из опытного Ан-72 для
эксплуатации в районах Арктики и Антарктиды под обозначением Ан-72А
«Арктический»</text>
<ngrams>
<ngram begin="0" end="48" nvalue="6" hashvalue="1186874997"/>
<ngram begin="6" end="56" nvalue="6" hashvalue="-1246273456"/>
...
</ngrams>
```

Рис. 11 Пример формата хранения документов

Подобное хранение позволяет обеспечить более высокую скорость поиска заимствований, чем при использовании необработанных документов, поскольку не требуется выполнять этапы нормализации и генерации шинглов.

3.4 Экспериментальная проверка

Для проверки качества построенного алгоритма и разработанной системы, необходимо провести тестирование. Рассмотрим методологию тестирования и различные тестовые выборки.

В качестве меры эффективности были использованы точность, полнота и F-мера. Значением полноты является отношение количества всех верно обнаруженных слов плагиата к количеству верных, т.е. размеченными как плагиат словах в тестовой выборке. Значением точности является количество всех верно обнаруженных слов плагиата к количеству обнаруженных слов плагиата. F-мера - это среднее гармоническое между точностью и полнотой. Тестирование производилось при различных настройках тестовой среды, а именно при различных константах алгоритма построения шинглов.

Для экспериментов было создано два тестовых набора. Первый тестовый набор был создан из материалов сообщества "Диссернет", содержащих размеченные на плагиат и корректные заимствования диссертации последних лет. Второй тестовый набор был создан вручную, так как первый тестовый набор содержит небольшое количество модификаций при заимствованиях, а в случае их наличия они не определяются как заимствования (подробнее в 3.4.1). Рассмотрим каждый из тестовых наборов по отдельности.

3.4.1 Экспериментальный тестовый набор на основе "Диссернет"

Этот тестовый набор был создан на основе материалов сообщества "Диссернет". Для этого тестового набора было выбрано 150 фрагментов текста из различных диссертаций, содержащих размеченный плагиат. В базу документов источников было отобрано 1000 соответствующих документов для поиска. Были произведены запуски с различными параметрами системы. В таблице 1 показаны результаты запусков с одинаковыми параметрами на фильтрацию документов и поиск заимствований, в таблице 2 - с различными параметрами.

Таблица 1. Результаты тестирование на выборке "Диссернет"

Длина шингла	Точность	Полнота	F-мера
Длина шингла N = 8	89,87%	46,40%	61,20%
Длина шингла N = 7	87,20%	52,12%	65,35%
Длина шингла N = 6	85,04%	62,74%	72,20%

Длина шингла N = 5	80,26%	66,32%	72,62%
Длина шингла N = 4	67.12%	75.16%	70,91%
Длина шингла N = 3	11.87%	94.11%	21,08%

Проанализируем полученные результаты для первой таблицы. Высокое значение точности при большой длине шингла объясняются тем, что при росте длины шингла увеличивается точность, так как вероятность нахождения в двух документах одинаковой фразы, не являющейся плагиатом уменьшается при увеличении длины фразы. Низкое значение полноты объясняется тем, что большое количество заимствований в диссертациях, используемых в тестовой выборке представляет собой относительно короткие фразы, длиной от 3 до 6 слов, которые не могут быть обнаружены при выбранной длине шингла, а также различными принципами выделения границ заимствования, описанными выше.

Дальнейшее увеличение длины шингла в тестах не приводит к существенному увеличению точности, но приводит к уменьшению значения полноты, так как большинство заимствованных фраз являются более короткими, чем 8 слов подряд.

Таблица 2. Результаты тестирование на выборке "Диссернет" с предварительной фильтрацией документов.

Длина шингла для фильтрации документов	**Длина шингла для сравнения документов**	**Точность**	**Полнота**	**F-мера**
N = 8	N = 3	84,87%	58.13%	68,99%
N = 8	N = 4	86,13%	52,11%	64,93%
N = 7	N = 3	79,04%	85,75%	82,25%
N = 7	N = 4	81,28%	68,31%	72,23%
N = 6	N = 3	65.12%	89.16%	75,26%
N = 6	N = 4	79.87%	71.11%	72,23%

Проанализируем полученные результаты из второй таблицы. Улучшение F-меры за счет использования двух этапов достигается по следующей причине: вероятность заимствования коротких фраз из документа сильно возрастает при факте наличия заимствования длинных фраз. Поэтому если в двух документах совпадает длинная фраза, то совпадения коротких фраз мы также считаем случаем заимствования. С помощью этого механизма происходит ликвидация большого количества ложных срабатываниях при анализе других документов.

Также проанализируем, почему даже при малой длине шингла не удается достичь 100 процентной полноты, а при большой длине - 100 процентной точности. Визуальный анализ тестовой выборки показал, что в данной тестовой выборке в качестве методов скрытия плагиата применяется в основном изменение порядка слов в предложении, в редких случаях замена символов (а именно символа "ё" на "é"), синонимизация и изменение окончаний. При этом "Диссернет" не обнаруживает факт заимствования при подобных методах скрытия факта заимствования (см. пример на рисунке 12, "Диссернет", [13]). Также "Диссернет" не обозначает плагиатом повторное заимствование фразы (см. пример на рисунке 13, "Диссернет"[13]: несмотря на очевидное заимствование слов "направлений экономической мысли" во фразе "были объектами исследований различных направлений экономической мысли" из аналогичной фразы в документе-источнике, эти слова не выделены как заимствованные, т.к. в предложении выше эти же самые слова обозначены как заимствованные).

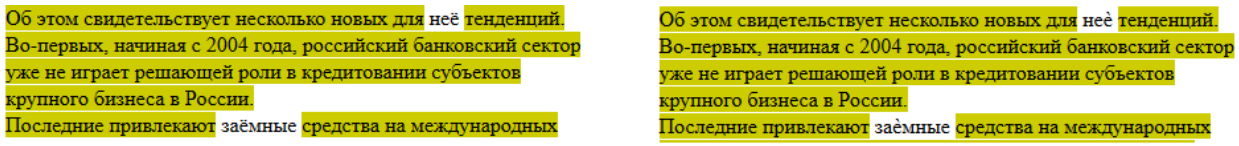

Рисунок 12. Пример выделения заимствований в случае замены символов

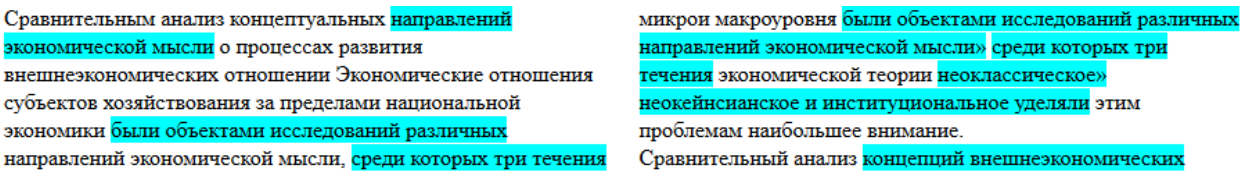

Рисунок 13. Пример ошибки выделения заимствования в наборе "Диссернет"

По итогам тестирования этого набора можно сделать вывод, что разработанное средство может применяться для анализа заимствований в диссертациях, так как для английского языка методы поиска плагиата дают результаты, близкие к полученным (согласно данным конкурса автоматических систем поиска плагиата [3]). Также высокие результаты F-меры показывают, что результат работы построенной программной системы

даёт близкие результаты к результатам разметки на плагиат сообществом "Диссернет". Однако, учитывая то, что в материалах "Диссернет" описанные методы скрытия заимствований применяются довольно редко и "Диссернет" не обозначает такие фразы как плагиат, требуется тестирование на других данных для проверки качества возможностей обнаружения скрытия плагиата.

3.4.2 Экспериментальный тестовый набор на основе новостей

Этот тестовый набор был создан вручную. В качестве основ для текстов использовались материалы новостных сайтов Lenta.ru, Яндекс.Новости, Газета.ру. Для них вручную воспроизводились случаи заимствований, при этом использовались техники скрытия заимствований, как по отдельности, так и в совокупности. Заимствования производилось только из одного документа-источника. Также было добавлено несколько текстов без заимствований для анализа ложных срабатываний (см. таблицу 3).

Таблица 3. Описание построенной выборки

Тип плагиата	**Количество статей**
Без плагиата	10
Плагиат без модификаций	10
С заменой русских символов на латинские	5
С заменой слов на синонимы	5
Изменение формата текста	5
С изменением родов, чисел	5
Перестановка, добавление и удаление слов	5
Использование совокупности методов	5
Всего:	50

Были проведены тесты при различных параметрах N для фильтрации и сравнения документов. В качестве возможных документов-источников рассматривались все документы-источники, использованные на этапе построения тестов. В итоге были получены следующие результаты (таблица 4).

Таблица 4. Результаты тестирования на новостной выборке.

Длина шингла для фильтрации документов	Длина шингла для сравнения документов	Точность	Полнота	F-мера
N = 8	N = 3	86,77%	63.31%	72,20%
N = 8	N = 4	90,23%	65,12%	75,64%
N = 7	N = 3	82,14%	88,65%	85,27%
N = 7	N = 4	84,38%	71,24%	77,25%
N = 6	N = 3	67.22%	94.32%	78,49%
N = 6	N = 4	81.77%	73.23%	77,26%

На таблицах видно, что при росте длины шингла увеличивается точность и уменьшается полнота.

Для демонстрации значимости методов борьбы модификации с заимствованиями были проведены подобные тесты с отключенными модулями борьбы с модификациями. Результаты показаны на таблице 5.

Таблица 5. Результаты тестирования на новостной выборке с отключенными модулями борьбы с модификациями.

Длина шингла для фильтрации документов	Длина шингла для сравнения документов	Точность	Полнота	F-мера
N = 8	N = 3	92,77%	16.31%	27,74%
N = 8	N = 4	96,23%	13,12%	23,09%
N = 7	N = 3	91,14%	26,65%	41,24%
N = 7	N = 4	93,38%	22,24%	35,86%
N = 6	N = 3	87.22%	34.32%	49,25%
N = 6	N = 4	85.77%	25.23%	38,99%

Как видно из результатов, использование модулей нормализации текста увеличивает полноту метода, но также увеличивается шанс ложного срабатывания, и как следствие, уменьшение точности.

Ошибки, приводящие к неверному результату работы алгоритма возникали на различных этапах работы программы, поэтому для повышения эффективности общей работы программы необходимо модернизировать каждый из отдельных модулей. Рассмотрим каждую из встречаемых ошибок по отдельности:

1**) Неверная десинонимизация**. Неверная десинонимизация одного слова в большинстве случаев приводила к тому, что фраза распознавалась как плагиат, однако конкретное измененное слово как плагиат не обозначалось, что приводило к снижению параметра Recall. Для борьбы с этой ошибкой необходимо добавить синоним в словарь.

2) **Неверное удаление стоп слов.** Некорректное удаление стоп слов приводит к ситуации, описанной в предыдущем пункте. Для борьбы с этой ошибкой необходимо расширить словарь стоп-слов.

3) **Отсутствие обнаружения из-за слишком короткого шингла.** Небольшой размер заимствованного текста, меньший, чем заданный размер шингла приводит к тому, что заимствование не обнаруживается. Для борьбы с этой ошибкой необходимо уменьшать длину шингла.

4) **Ложные срабатывания.** Ложные срабатывания происходили по нескольким причинам. В большинстве случаев это общие фразы, используемые для связи слов в предложении, либо вводные фразы. Для борьбы с этой проблемой нужно расширять словарь стоп-слов и увеличивать длину шингла.

Как видно из описания решения проблем 3 и 4, необходимо выбирать определенное значение длины шингла, чтобы поддерживать баланс между ошибками 3 и 4. Конкретное значение нужно выбирать в зависимости от предметной области, в случае описанной тестовой выборке значение можно выбрать на основе таблиц выше.

По итогам тестирования данной выборки можно сделать вывод, что описанные методы борьбы с модификациями действительно позволяют обнаруживать модифицированные заимствования с высокими показателями точности и полноты.

4 Описание практической части

Для создания графического интерфейса программы была выбрана Java библиотека Swing. Было решено использовать эту библиотеку, так как Swing относительно прост для создания интерфейса и не требовал значительных временных затрат на разработку при отсутствии требований к графическому интерфейсу. Swing был разработан компанией Sun Microsystems.

В качестве одной из реализаций разбиения текста на предложения использована библиотека OpenNLP. Библиотека OpenNLP предоставляет широкий спектр инструментов для обработки текстов, разработана компанией Apache.

4.1 Общая схема работы

Диаграмма на рисунке 14 показывает общую схему работы разработанного программного средства.

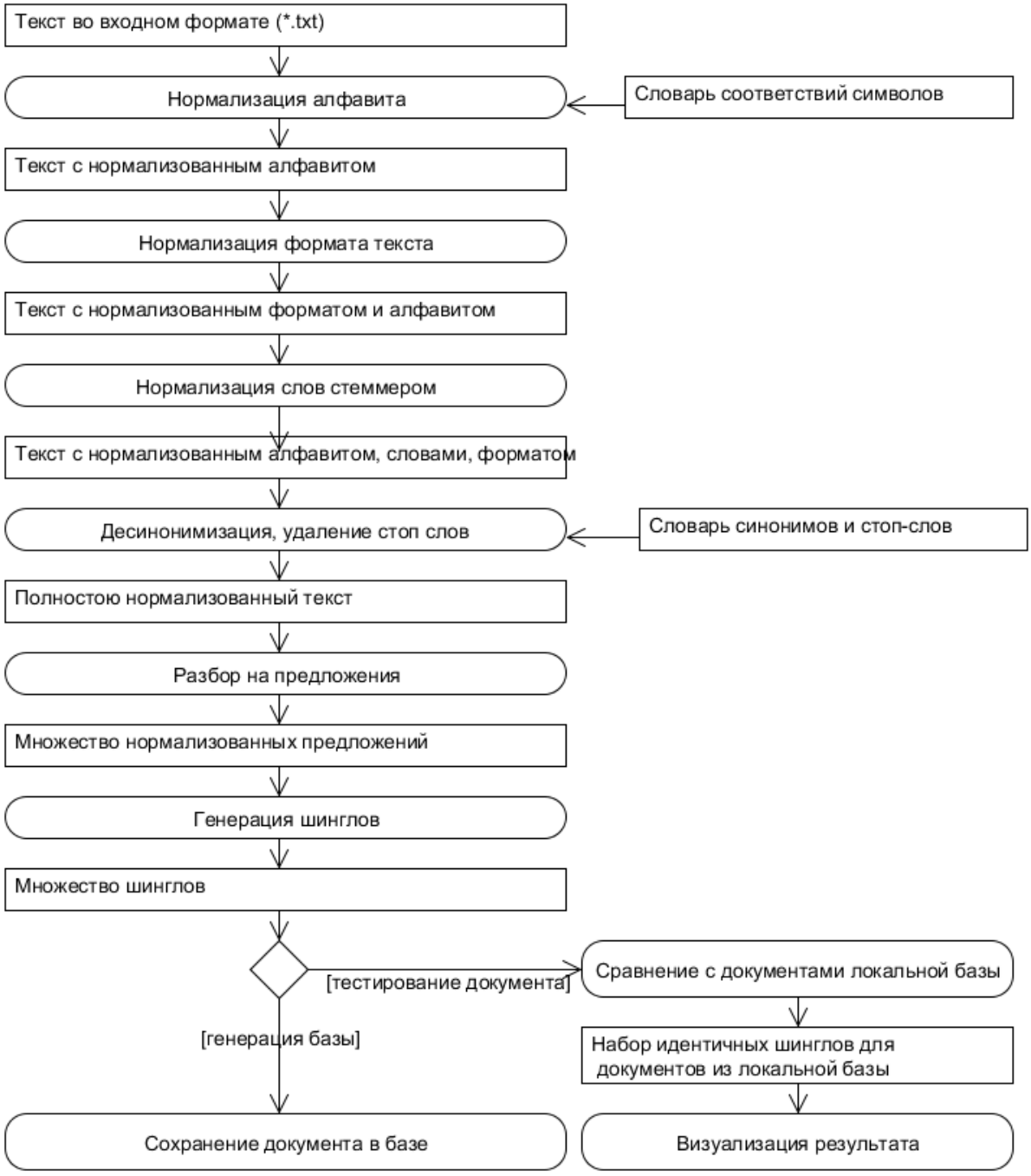

Рисунок 14. Общая схема работы

Для нормализации алфавита, десинонимизации, удаления стоп слов используются соответствующие словари, загружаемые с жесткого диска при старте программы. При нормализации происходит сравнение нормализуемого фрагмента (слова или символа, в зависимости от этапа), и в случае совпадения происходит замена или удаление символа/слова.

Стеммер Портера реализован с помощью регулярных выражений для поиска соответствующих суффиксов.

Для разбиения исходного текста на предложения используется два алгоритма: разбиение по знакам препинания, таким как точка, восклицательный знак, вопросительный знак, и реализация алгоритма Apache OpenNLP Sentence Detector, основанный на машинном обучении.

Для получения хеш кодов шинглов используется стандартный метод hashCode() получения хеш кода для строк класса String.

Сортировка слов в предложениях осуществляется с помощью стандартных методов сортировки Java Collections.Sort и стандартной функции сравнения строк String.Compare().

Для каждого из документов локальной базы создаётся отдельный текстовый файл, формат которого описан в пункте 3.3.

Пример визуализации работы предоставлен в приложении А. Окно визуализатора содержит два текстовых поля: первое (слева) для ввода проверяемого текста, второе для отображения текста-источника. После проверки текста на заимствования с помощью кнопок в центре можно выбрать один из документов-источников, и в текстах на правом и левом текстовых полях будут выделены заимствованные наборы слов.

4.2 Общая архитектура системы

На рисунке 15 представлена диаграмма классов, показывающая общую архитектуру системы.

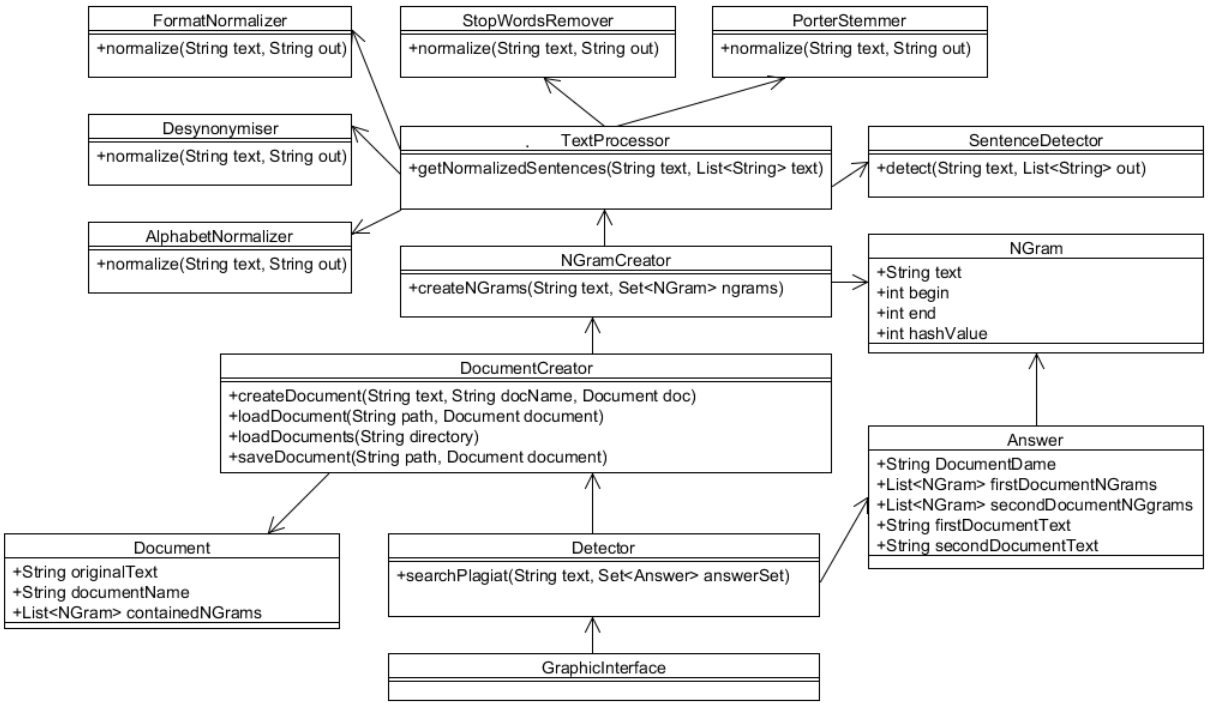

Рисунок 15. Диаграмма основных классов системы

Класс *GraphicInterface* является классом, отвечающим за визуализацию и взаимодействие с пользователем.

Класс *Detector* является основным классом, отвечающим за поиск заимствований в данном на обработку тексте.

Класс *DocumentCreator* является классом, отвечающий за создание размеченных на шинглы документов, сохранением их в локальной базе и загрузки документов из локальной базы.

Класс *NGramDetector* является классом, создающим шинглы для данного текста.

Класс *TextProcessor* является классом, производящим нормализацию текста и разбиение его на предложения. Для этих целей класс использует классы *WordSorter*, *Desynonymiser*, *FormatNormalizer*, *StopWordsRemover*, *AlphabetNormalizer*, *PorterStemmer* для сортировки слов, десинонимизации, нормализации формата, удаления стопслов, нормализации алфавита и стемминга соответственно. Подробно эти этапы описаны в разделах 4.1 и 3.1.

Класс *Answer* является классом для хранения данных, необходимых для визуализации ответа.

Класс *NGram* является классом для хранения данных, представляет собой сущность "шингл".

Класс *Document* является классом для хранения данных, представляет собой сущность "размеченный документ".

4.3. Характеристики функционирования

Время анализа одного документа напрямую зависит от количества документов в локальной базе, так необходимо произвести поиск шинглов документа во всех документах локальной базы. Поэтому измерения производительности проводились в два этапа:

1) измерение производительности генерации шинглов для документа

2) измерение производительности сравнения шинглов исходного документа и документа-источника.

Для измерения производительности было отобрано 300 документов для проверки и 1000 документов в локальной базе.

Для определения времени генерации шинглов для было произведено 3 тестовых запуска на 1000 документах, посчитано среднее арифметическое между ними. Для генерации использовались параметры N=6, M=1.

Полученные значения: 198с, 197с, 198с

Соответственно, среднее время для генерации шинглов для одного документа составляет: 198мс.

Для определения времени сравнения шинглов двух документов было определено время сравнения для 300 размеченных документов с 1000 размеченными документами: 3997с, 3995с, 3999с. Для сравнения использовались шинглы с аналогичными параметрами N=6, M=1.

Соответственно, среднее время сравнения шинглов двух документов составляет: 13мс. Следовательно, для того, чтобы проверить один документ, необходимо примерное время

T = (198 + 13*N) мс, где N - число документов в локальной базе.

Измерения были проведены при конфигурации тестового компьютера: Intel Core2Duo (2 Core) 3.0 GHz, 4096 Mb RAM.

Заключение

В рамках дипломной работы были получены следующие результаты:

1. Исследованы существующие методы поиски заимствований.

2. Исследованы существующие методы скрытия заимствований, а также методы борьбы с ними.

3. Разработан метод, позволяющий производить поиск заимствований в условиях скрытия факта заимствования.

4. Создана программная система, позволяющая производить поиск заимствований в русскоязычных текстах на основе разработанного метода с возможностями визуализации результата, сохранения нормализованных документов в локальной базе и автоматического распознавания модифицированных заимствований.

5. Созданы тестовые выборки, а также проведено тестирование эффективности разработанной системы в терминах точности и полноты и F-меры, а также производительности.

Список литературы

1. Зеленков Ю. Г. и Сегалович И. В. Сравнительный анализ методов определения нечетких дубликатов для Web-документов // Труды 9ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL’2007, Переславль-Залесский, Россия, 2007.
2. Soledad Pera Maria и Ng Yiu-Kai A Structural, Content-Similarity Measure for Detecting Spam Documents on the Web // INTERNATIONAL JOURNAL OF WEB INFORMATION SYSTEMS, С. 431-464
3. Potthast Martin Overview of the 4th International Competition on Plagiarism Detection // CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, Rome, 2012.
4. Barrŕon-Cede˜no Alberto и Rosso Paolo On Automatic Plagiarism Detection Based on n-Grams Comparison Advances in Information Retrieval // Lecture Notes in Computer Science Volume 5478, 2009, С. 696-700
5. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. *//Вильямс, 2012*, С. 60-70
6. Ilyinsky Sergey An efficient method to detect duplicates of Web documents with the use of inverted index // WWW'2002 — Eleventh International World Wide Web Conference, 2002, С.220-228
7. Osama Haggag and Samhaa El-Beltagy Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring // PAN at CLEF 2013, 2013
8. Diego A. Rodríguez Torrejón, José Manuel Martín Ramos Text Alignment Module in CoReMo 2.1 Plagiarism Detector Notebook for // PAN at CLEF 2013, 2013
9. *A. Broder.* On the resemblance and containment of documents. Compression and Complexity of Sequences // SEQUENCES'97, IEEE Computer Society, 1998, С. 21-29
10. *D. Fetterly, M. Manasse, M. Najork.* A Large-Scale Study of the Evolution of Web Pages // WWW2003, 2003, Budapest, Hungary.
11. *Шарапов Р. В., Шарапова Е. В.* Исследование плагиата в работах студентов // Сборник «Компьютерная лингвистика и интеллектуальные технологии», 2012, С. 578-587
12. *David Eppstein* Longest Common Subsequences // Journal of Information Processing., 1990, С.463–469.
13. Сообщество "Диссернет" Экспертиза диссертации Понятовской Анны Федоровны // "Диссернет", 2014, [HTML] (wiki.dessernet.org/w/PonatovkayaAF2010.html)

14. A. Chowdhury, O. Frieder, D. Grossman, M. McCabe. Collection statistics for fast duplicate document detection // ACM Transactions on Information Systems (TOIS, 2002

15. El-Beltagy, S.R., Rafea, A.: KP-Miner: A keyphrase extraction system for English and Arabic documents // Information Systems, 2009, С. 132–144.

16. Jones K. S. A statistical interpretation of term specificity and its application in retrieval // Journal of Documentation : MCB University: MCB University Press, 2004. С. 493-502.

17. Porter M. F. Algorithm for suffix stripping // Program, 1980, С. 130—137.

18. А. Коваленко. Вероятностный морфологический анализатор русского и украинского языков» , 2002, [HTML] (http://www.keva.ru/stemka/stemka.html)

19. Ilya Segalovich. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine, 1998, [PDF] (http://download.yandex.ru/company/iseg-las-vegas.pdf)

20. Абрамов Н. Словарь русских синонимов и сходных по смыслу выражений // М.: Русские словари, 1999.

Приложение А.

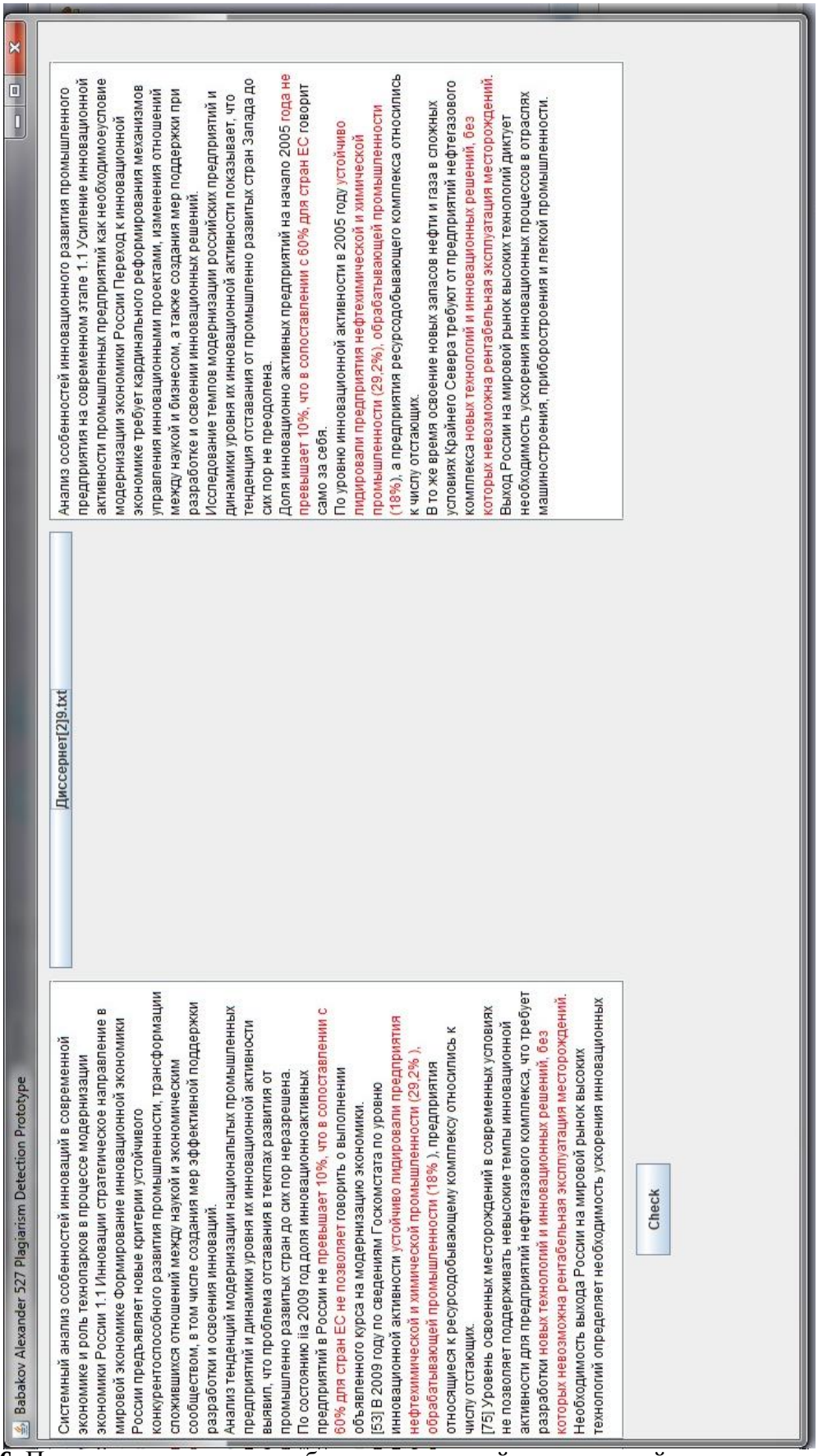

Рисунок 16. Пример визуализации работы построенной программной системы.