Сервисы полнотекстового поиска в информационно-аналитической системе (Часть 1)[1]

И.В. Соченков, Р.Е. Суворов

Аннотация: Настоящее исследование посвящено разработке структур данных и алгоритмов информационного поиска. Статья представляет первую часть исследования и содержит описание разработанных структур данных для представления поисковых индексов в информационно-аналитической системе. Предлагаемые структуры данных ориентированы на представление лингвистической информации текстов, включая синтаксическую и семантическую. В статье рассмотрены алгоритмы построения поисковых индексов.

Ключевые слова: информационный поиск, семантический поиск, полнотекстовый поиск, структуры данных для информационного поиска, инвертированный индекс, информационно-аналитическая система.

Введение

Сервисы полнотекстового поиска являются неотъемлемой частью любой современной информационно-аналитической системы (ИАС). Они составляют основу информационного обеспечения ИАС и предоставляют экспертам-пользователям доступ к интересующей информации.

Поисковые сервисы ИАС должны реализовывать различные режимы поиска: по ключевым словам и терминам предметных областей, фразовый, семантический и вопросно-ответный [1, 2]. При этом должна учитываться зона поиска: ограничения на интересующие пользователя документы по различным метаданным.

Помимо перечисленных требований методы полнотекстового поиска должны поддерживать обработку больших массивов информации, а их реализация должна быть вычислительно эффективной для обеспечения аналитических функций ИАС, таких как, например, анализ динамики публикаций в некоторой области.

Целью исследования является создание алгоритмического обеспечения сервисов ИАС, позволяющего реализовать качественный и эффективный полнотекстовый поиск и анализ текстовой информации. Задачи исследования состоят в разработке алгоритмов и структур данных для полнотекстового поиска, интегрированных в ИАС, а также в экспериментальной проверке предложенных алгоритмов и оценке качества их работы.

Основу предлагаемых алгоритмов составляет модель текста, являющаяся развитием реляционно-ситуационной модели Г.С. Осипова – Г.А. Золотовой [3], и метод сравнения текстов, предложенные в работе [4]. Разработанные алгоритмы и структуры данных, предназначенные для обеспечения функций полнотекстового поиска, учитывают лингвистическую информацию, выделяемую из текстов автоматически на различных уровнях анализа: лексико-морфологическом, синтаксическом и семантическом.

[1] Работа выполнена при финансовой поддержке Минобрнауки России по Государственному контракту № 14.514.11.4018 в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы».

Представляемое алгоритмическое обеспечение ИАС для реализации сервисов полнотекстового поиска является результатом многолетних исследований в области методов построения поисковых индексов и ранжирования результатов поиска, выполненных И.В. Соченковым под руководством Г.С. Осипова в коллективе исследователей проекта Exactus [3, 5, 6]. Экспериментальная проверка исследуемых алгоритмов и оценка качества результатов поиска выполнена Р.Е. Суворовым.

Настоящая статья представляет первую часть исследования и содержит описание предложенных структур данных для решения задач полнотекстового поиска, а также рассмотрение разработанного метода построения поискового индекса. Разработанные структуры данных поисковых индексов предназначены для хранения и организации доступа к лингвистической информации текстов естественного языка, включая синтаксические и семантические связи.

Вторая часть исследования посвящена разработке алгоритмов оценки релевантности и ранжирования результатов информационного поиска. Предлагаемые методы лежат в основе поисково-аналитических сервисов ИАС, которые реализуют различные режимы поиска. Также во второй части рассматриваются вопросы экспериментальной проверки и практического применения созданного алгоритмического обеспечения в ИАС.

1. Структуры данных для полнотекстового поиска

1.1. Представление документов

Для представления информации об отдельных документах в подсистеме полнотекстового поиска ИАС рассмотрим структуру данных, содержащую:

– нетекстовую метаинформацию документов (например, идентификатор документа, дату публикации, коды тематических классов, формат документа и др.)

– индекс текста документа (ИТД), построенный с помощью лингвистического анализа.

ИТД представляет собой последовательность элементов данных (ЭД), в которой каждый ЭД однозначно соответствует вхождению некоторого слова в текст документа. Каждый ЭД содержит следующий набор полей данных:

1. идентификатор нормальной формы лексемы ИНФЛ (word_id), соответствующей вхождению слова в текст документа;

2. тег метаинформации, к которой относится вхождение слова в тексте документа (tag_id);

3. идентификатор формы (ИФ) вхождения слова в текст запроса (form);

4. разность между позицией слова, представляемого ЭД, и позицией слова, являющегося главным элементом синтаксической группы (ГЭСГ), в которую входит рассматриваемое вхождение слова в тексте документа; поле имеет нулевое значение, если вхождение слова само является ГЭСГ;

5. множество значений синтаксем (если эти значения приписаны вхождению слова в тексте);

6. множество значений синтаксем, связанных семантической связью в тексте документа с текущим вхождением слова в том же предложении;

7. вес вхождения – lTF в соответствии с работой [4];

8. признак наличия у вхождения слова зависимых слов в тексте, относящихся к синтаксической категории определения (атрибута).

Предложенные структуры данных представляют лексико-морфологическую, синтаксическую и семантическую информацию текста: в ИТД хранится не только информация, характеризующая отдельные вхождения слов в тексте, но и синтаксические и семантические связи между ними.

1.2. Инвертированный поисковый семантический индекс

Для эффективного поиска информации в коллекциях документов применяется инвертированный поисковый индекс (ИПИ) [7]. Разработанный в рамках исследования ИПИ ориентирован на хранение информации о текстах масштабных коллекций текстовых документов. Отличительной особенностью семантического ИПИ является эффективное хранение и выборка информации о текстах, включая синтаксическую и семантическую информацию [8, 9].

ИПИ позволяет для каждого слова запроса пользователя получить список вхождений этого

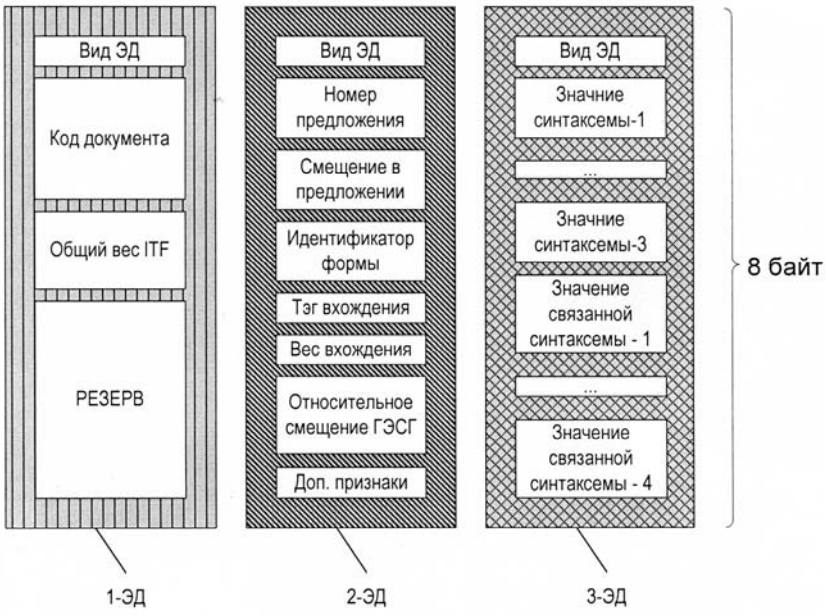

Рис. 1. ЭД разработанного инвертированного поискового индекса

слова в документы коллекции и определить соответствующую этим вхождениям лингвистическую информацию. ИПИ обычно реализуется в виде хеш-таблицы. В качестве ключа ИПИ выступает ИНФЛ. Список вхождений соответствующего слова в документы коллекции представляет собой массив ЭД различных видов (обозначим их σ-ЭД). Каждый σ-ЭД занимает фиксированный размер в памяти и содержит набор полей данных, различающийся у σ-ЭД в зависимости от вида.

Рассмотрим различные виды σ-ЭД. ЭД метаинформации документа (1-ЭД) содержит следующие поля данных:

1) вид ЭД (1);

2) код документа в индексной базе, предназначенный для идентификации документа в ИПИ (doc_id);

3) суммарный вес всех вхождений слова в соответствующий документ.

σ-ЭД вхождения слова (2-ЭД) содержит информацию, которая определяется для каждого вхождения слова в тексты проиндексированных документов:

1) вид ЭД (2);

2) поля 2–4 и 7, 8 ЭД ИТД.

Семантический σ-ЭД (3-ЭД) содержит информацию, которая определяется для некоторых вхождений слов и при этом может отсутствовать у других. Семантический маркер вхождения слова содержит следующие поля:

1) вид ЭД (3);

2) поля 5, 6 ЭД ИТД.

Рассмотрим более подробно предложенную схему представления информации в ИПИ (Рис. 1).

Заметим, что семантическая информация, связанная с вхождением слова в текст, размещается в семантической структуре данных размером 8 байт. Каждое значение синтаксемы представляется числом, занимающим 7 бит. При таком подходе для вхождения слова сохраняется до 3 семантических значений и до 5 значений связанных синтаксем (*(3+5)x7=56* бит) в сложном предложении [4].

По сравнению с классической схемой хранения информации о вхождениях слов, в которой на каждое вхождение приходится отдельный ЭД (Рис. 2), содержащий полный набор полей данных, предложенная схема представляет информацию о последовательности вхождений некоторой лексемы в текст последовательностью σ-ЭД. Последовательность σ-ЭД, относящихся к вхождениям некоторой лексемы в текст одного документа, назовём σ-цепочкой; σ-цепочка строится по следующим правилам:

1) в начале цепочки размещается 1-ЭД;

2) за ним следует 2-ЭД, соответствующий первому (по порядку) вхождению слова в текст;

3) за ним опционально следует 3-ЭД, если с рассматриваемым вхождением слова связана некоторая семантическая информация;

4) далее следуют чередующиеся 2-ЭД и 3-ЭД (опционально), соответствующие остальным вхождениям слов.

Для представления информации о вхождениях слов во все документы коллекции применяется последовательность σ-цепочек – σ-последовательность. σ-последовательность формируется с помощью конкатенации σ-цепочек по следующему правилу: σ-цепочка σ_1 предшествует σ_2 тогда и только тогда, когда код документа, представляемого σ-цепочкой σ_1 меньше кода документа, представляемого σ-цепочкой σ_2.

Таким образом на классе σ-цепочек индуцируется линейный порядок (*σ-правило-1*): σ-цепочка, в начале которой находится 1-ЭД с кодом документа d_1 предшествует σ-цепочке, в начале которой находится 1-ЭД с кодом документа d_2, тогда и только тогда, когда $d_1<d_1$. Аналогичным образом определяется частичный порядок на классе σ-ЭД внутри σ цепочки (*σ-правило-2)*:

1) 2-ЭД m_1 предшествует другому 2-ЭД m_2 тогда и только тогда, когда

a) номер предложения в тексте документа, соответствующий вхождению слова, представленного 2-ЭД m_1, меньше номера предложения в тексте документа, соответствующего вхождению слова, представленного 2-ЭД m_2: $sent(m_1) < sent(m_2)$;

b) номера предложений совпадают ($sent(m_1) = sent(m_2)$), а смещение (от начала предложения) слова, представленного 2-ЭД m_1, меньше смещения слова, представленного 2- ЭД m_2: $sofs(m_1) < sofs(m_2)$;

2) 2-ЭД m_1 предшествует 3-ЭД m_1', содержащему семантическую информацию о вхождении того же слова, что и 2-ЭД. В общем случае 2-ЭД и 3-ЭД, соответствующие разным вхождениям слов, не сравнимы.

В совокупности *σ-правило-1* и *σ-правило-2* вводят отношение частичного порядка на классе σ- ЭД в σ-последовательностях (Рис. 3).

Заметим, что введённое отношение частичного порядка (*σ-правило-2*) не позволяет подвергать слиянию произвольные σ-последовательности. Для устранения этого недостатка рассмотрим следующее *σ-правило-3*: из двух

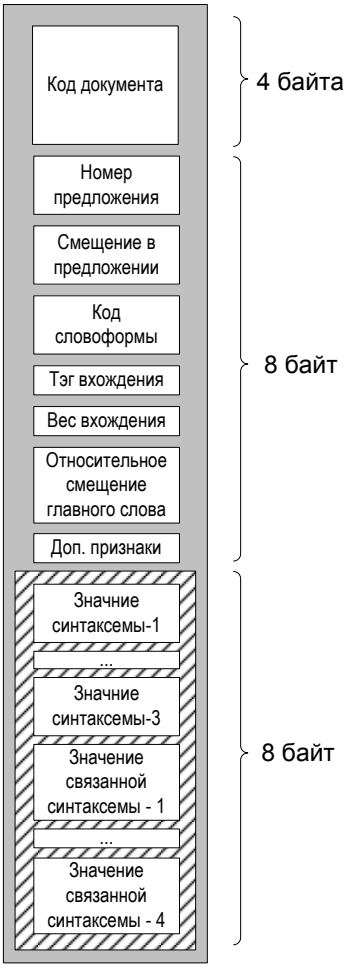

Рис.2. Элемент данных в классическом инвертированном поисковом индексе

сравниваемых σ-ЭД m_1 и m_2 ЭД m_1 предшествует m_2 ($m_1<m_2$) тогда и только тогда, когда:

1) m_1 является 3-ЭД, а m_2 – 2-ЭД;

2) m_1 является 3-ЭД, а m_2 – 1-ЭД;

3) m_1 является 2-ЭД, а m_2 – 1-ЭД;

4) m_1 и m_2 являются 1-ЭД, и при этом код документа в m_1 меньше кода документа в m_2: $doc\_id(m_1) < doc\_id(m_2)$;

5) m_1 и m_2 являются 2-ЭД, и при этом

a) номер предложения в тексте документа, соответствующий вхождению слова, представляемого 2-ЭД m_1, меньше номера предложения в тексте документа, соответствующего вхождению слова, представляемого 2-ЭД m_2: $sent(m_1) < sent(m_2)$;

b) номера предложений совпадают ($sent(m_1) = sent(m_2)$), а смещение (от начала

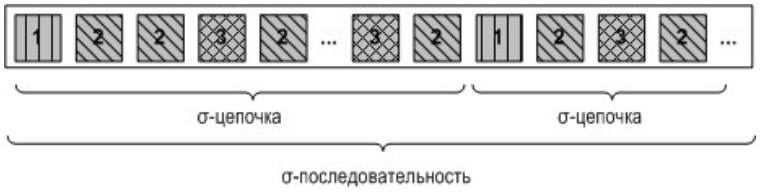

Рис. 3. Пример упорядочения ЭД в σ-последовательностях

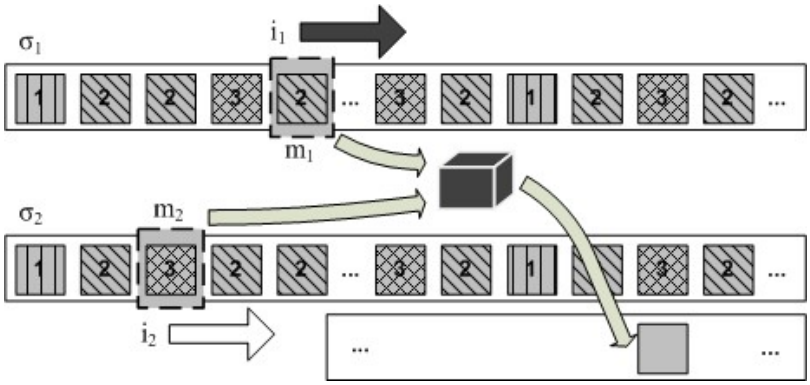

Рис. 4. Пример обработки σ-последовательностей

предложения) слова, представляемого 2-ЭД m_1, меньше смещения слова, представляемого 2- ЭД m_2: $sofs(m_1) < sofs(m_2)$.

Отметим, что *σ-правило-3* задаёт отношение частичного порядка: рассмотрены 5 вариантов сравнения σ-ЭД из 6 возможных, причём пара 3-ЭД не сравнима между собой.

Рассмотрим класс алгоритмов *S*, обладающих следующими свойствами:

1. алгоритмы класса *S* последовательно обрабатывают две σ-последовательности слева направо и используют *σ-правило-3* в качестве отношения порядка для сравнения ЭД;

2. на каждом шаге своей работы алгоритмы класса *S* перемещают тот итератор по σ-последовательности, который указывает на меньший ЭД из сравниваемых (в смысле *σ-правила-3*) (Рис. 4).

Покажем, что необходимость сравнения 3-ЭД между собой не возникает на классе алгоритмов *S*. Пусть в какой-то момент *t* работы алгоритма *a* из указанного класса *S* для сравнения выбираются σ-ЭД m_1 и m_2 из обрабатываемых σ-последовательностей σ_1 и σ_2 соответственно. Пусть m_1 и m_2 оба являются 3-ЭД. Тогда они находятся не в начале σ-последовательностей σ_1 и σ_2 (поскольку в противном случае последовательности σ_1 и σ_2 не удовлетворяли бы определению σ-последовательности, что противоречит условию). Это означает, что итераторы, указывающие на m_1 и m_2 в σ_1 и σ_2 соответственно (i_1 и i_2), были перемещены в ходе предыдущих шагов работы алгоритма *a*, т.е. $t \neq t_0$. Тогда на предыдущем шаге (*t-1*) работы алгоритма был перемещён, например, итератор по $\sigma_1 - i_1$, т.е. сравнивались σ-ЭД $Prev(m_1)$ и m_2. По определению σ-последовательности предыдущий элемент – $Prev(m_1)$ является 2-ЭД, т.е. на шаге *t-1* работы алгоритма *a* итератор i_1 последовательности σ_1 указывал на 2-ЭД ($Prev(m_1)$), предшествующий 3-ЭД (m_2). Значит, на шаге *t-1* сравнивались 2-ЭД и 3-ЭД. Но по п.1 *σ-правила-3* 3-ЭД предшествует 2-ЭД, т.е. $m_2 < Prev(m_1)$. По свойству алгоритмов класса *S* в этом случае должен быть перемещён итератор по $\sigma_2 - i_2$ (причём по определению σ-последовательности на шаге *t* i_2 будет указывать на следующий за 3-ЭД 2-ЭД или 1-ЭД). Полученное противоречие доказывает утверждение.

Как будет показано в следующем разделе, введённые отношения частичных порядков позволяют реализовать алгоритмы построения

σ-последовательностей для представления ИТД (используя *σ-правило-2)* и слияния σ-последовательностей (с помощью *σ-правила-3*).

Перейдём к оценке количества памяти, необходимого для хранения информации в ИПИ. В реализованной в ИАС схеме представления данных каждый σ-ЭД (независимо от вида) занимает в памяти 8 байт. При классической схеме представления списков вхождений в ИПИ («одно вхождение – один ЭД») каждый ЭД требует 20 байт для хранения всей требуемой информации: код документа – 4 байта, 8 байт – общая информация, связанная с вхождением слова, 8 байт – семантическая информация. На практике в научно-технических документах каждая лексема встречается в тексте в среднем около 3 раз, а доля вхождений слов с семантической информацией составляет 25–40% (поскольку семантические значения устанавливаются для синтаксем, являющихся главными элементами именных групп [9]). Из этих данных следует, что для хранения информации обо всех вхождениях некоторой лексемы в документ потребуется (Рис. 1, 2):

– в классической схеме организации ИПИ без учёта семантики – *3x12=36* байт;

– в классической схеме организации ИПИ с учётом семантики – *3x20=60* байт;

– в предложенном ИПИ без учёта семантики – *8+3x8=32* байт;

– в предложенном ИПИ с учётом семантики (в среднем) – *8+3x(8+8x0.4)=41.6* байт.

Соотношение размеров ИПИ при различном способе представлении информации приведено на диаграмме (Рис. 5).

Из приведённого сравнения следует, что предложенный ИПИ эффективно представляет информацию о вхождениях слов в тексты документов коллекции.

Далее перейдем к описанию разработанных алгоритмов для работы с представленными структурами данных.

2. Построение поисковых индексов

Процедура построения поисковых индексов заключается в лингвистическом анализе текстов, составляющих коллекцию, построении ИТД и помещении ИТД в ИПИ.

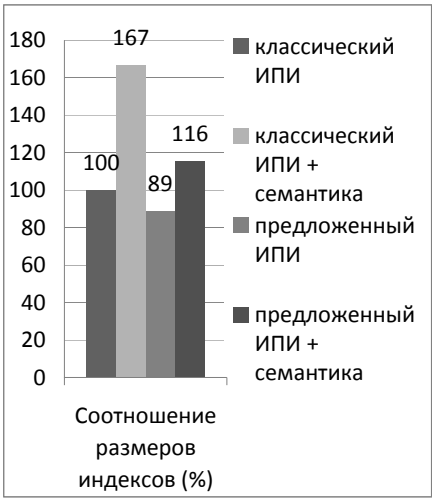

Рис. 5. Соотношение размеров инвертированных поисковых индексов при различных схемах представления информации

На этапе лингвистического анализа автоматически устанавливается информация, связанная с вхождениями слов [6, 8, 9]. Построенные ИТД передаются в систему наполнения ИПИ. Поскольку текст документа может содержать значительное количество различных лексем (от сотен до десятков тысяч), добавление ИТД к ИПИ требует обновления большого количества σ-последовательностей, соответствующих этим лексемам.

Обновление σ-последовательности состоит в слиянии исходных данных σ-последовательности с σ-цепочкой, содержащей информацию о вхождениях слова в текст обрабатываемого документа. В простейшем случае, если документы не удаляются из ИПИ, их коды строго возрастают, и ИПИ целиком размещается в оперативной памяти сервера ИАС, то слияние заключается в конкатенации исходной σ-последовательности с добавляемой σ-цепочкой (путём дописывания σ-цепочки в конец σ-последовательности). Эта операция корректна (не нарушает порядка в σ-последовательности) и на практике достаточно эффективна, если реализовать σ-последовательность как массив ЭД с предварительным резервированием.

В более сложном случае, ИПИ размещается в файловой системе и представляет собой хранилище данных «ключ-значение» (key-value storage) [10]. При этом может потребоваться

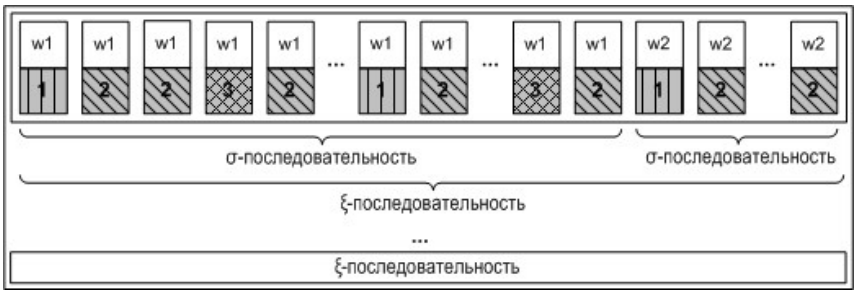

Рис.6. Пример упорядочения ЭД в ξ-последовательностях

удалять документы из ИПИ (например, по причине их устаревания). В этом случае алгоритм обновления включает в себя стадии считывания с диска и восстановления σ-последовательности из сжатого образа, слияния с σ-цепочкой, сжатия и записи на диск. Эти операции достаточно трудоёмки и при больших объёмах информации в ИПИ могут выполняться несколько секунд. Общее время обработки одного ИТД может достигать десятков минут. Поэтому для эффективной обработки поступающих ИТД выполняется буферизация. Структуру данных для промежуточного накопления ИТД назовём буфером ξ-последовательностей.

Буфер ξ-последовательностей представляет собой массив ξ-последовательностей. Каждый ЭД ξ-последовательности (ξ-ЭД) представляет собой пару <*идентификатор лексемы, σ-ЭД*> (Рис. 6). ЭД в ξ-последовательности частично упорядочены (ξ-*правило*): ξ-ЭД-1 предшествует ξ-ЭД-2 тогда и только тогда, когда:

А) идентификатор лексемы ξ-ЭД-1 меньше идентификатора лексемы ξ-ЭД-2;

Б) идентификаторы лексем ξ-ЭД-1 и ξ-ЭД-2 совпадают, а σ-ЭД из ξ-ЭД-1 предшествует σ-ЭД из ξ-ЭД-2 (в смысле *σ-правила-3*).

Таким образом, при добавлении нового документа в буфер ξ-последовательностей ИТД сортируется в соответствии со следующим правилом: ЭД ИТД μ_1 предшествует μ_2 тогда и только тогда, когда:

1) ИНФЛ μ_1 меньше ИНФЛ μ_1 ($word\_id(\mu_1) < word\_id(\mu_2)$);

2) ИНФЛ μ_1 совпадает с ИНФЛ μ_1 ($word\_id(\mu_1) = word\_id(\mu_2)$), но при этом выполнено:

a) номер предложения в тексте документа, соответствующий вхождению слова, представляемого μ_1, меньше номера предложения в тексте документа, соответствующего вхождению слова, представляемого μ_2: $sent(\mu_1) < sent(\mu_2)$;

b) при совпадении номеров предложений $sent(\mu_1) = sent(\mu_1)$ μ_1 смещение (от начала предложения) слова, представляемого μ_1, меньше смещения слова, представляемого μ_2: $sofs(\mu_1) < sofs(\mu_2)$.

На следующем этапе отсортированный ИТД преобразуется в ξ-последовательность, которая подвергается слиянию с ξ-последовательностью из буфера, имеющей наименьший размер.

Процедура слияния двух произвольных ξ-последовательностей выполняется с помощью классического алгоритма merge [7] (Рис. 7) с отношением порядка (<), введённым ξ-*правилом*. Как видно из блок-схемы, этот алгоритм относится к классу алгоритмов *S*, что обеспечивает корректность его применения для решения указанной задачи. Отметим, что результирующая последовательность также является ξ-последовательностью, поскольку в двух ξ-последовательностях, подвергаемых процедуре слияния, отсутствуют *конфликтные* ЭД (т.е. такие ЭД μ_1, μ_2, что при применении алгоритма из класса *S* они являются эквивалентными, это значит. что для них одновременно не выполнено $\mu_1 < \mu_2$ и $\mu_2 < \mu_1$ в смысле выбранного отношения порядка).

Оценим трудоёмкость представленного алгоритма помещения ИТД в буфер. Оценка сложности шагов алгоритма относительно количества вхождений слов (*N*) составляет:

- сортировка: $O(N \cdot log(N))$;
- преобразование в ξ-последовательность: $O(N)$;
- слияние ξ-последовательностей: $O(N + M)$,

где *M* – длина самой короткой ξ-последовательности в буфере.

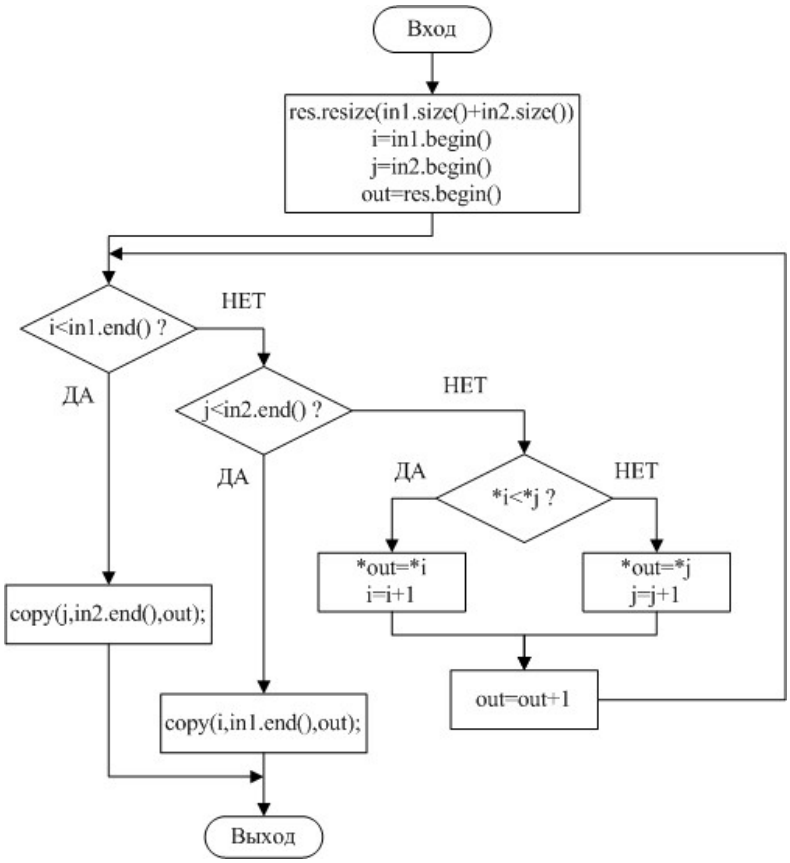

Рис. 7. Блок-схема алгоритма слияния последовательностей

Если после обработки очередного ИТД количество ЭД в буфере ξ-последовательностей превышает максимально заданное значение, выполняется объединение данных буфера с ИПИ. Для этого сначала выполняется слияние всех ξ-последовательностей в одну с помощью модифицированного алгоритма сбалансированного n-путевого слияния (n-ПС) на основе кучи (n-way merge) [11] с отношением порядка, заданным *ξ-правилом*. Затем с помощью последовательного прохода по полученной ξ-последовательности в ней выделяются σ-последовательности (Рис. 6), которые подвергаются слиянию с σ-последовательностями из ИПИ с соответственными ИНФЛ. Процедура слияния выполняется с помощью классического алгоритма merge[7] с *σ-правилом-3* в качестве отношения порядка.

При необходимости из σ-последовательности, полученной из ИПИ, могут быть исключены ЭД, соответствующие документам, коды которых присутствуют в списке удалённых документов. При этом удаляются все σ-цепочки, соответствующие удалённым документам. Предполагается, что список кодов удалённых документов отсортирован по возрастанию кодов. Блок-схема алгоритма фильтрации σ-последовательности путём удаления σ-цепочек представлена на Рис. 8. Указанная процедура фильтрации позволяет исключить из σ-последовательностей конфликтные элементы, поэтому в результате их слияния получается σ-последовательность.

Для шагов представленного алгоритма обновления ИПИ оценка сложности составляет:

- n-путевое слияние: $O(K \cdot log(N))$, где K – количество ξ-последовательностей, а N – количество ξ-ЭД в буфере;
- фильтрация σ-последовательностей: $O(P \cdot L)$, где P – количество различных лексем с различными ИНФЛ в ИПИ, L – средняя длина σ-последовательности в ИПИ;
- слияние σ-последовательностей: $O(P \cdot (M + L))$, где L – средняя длина σ-последовательности в ИПИ.

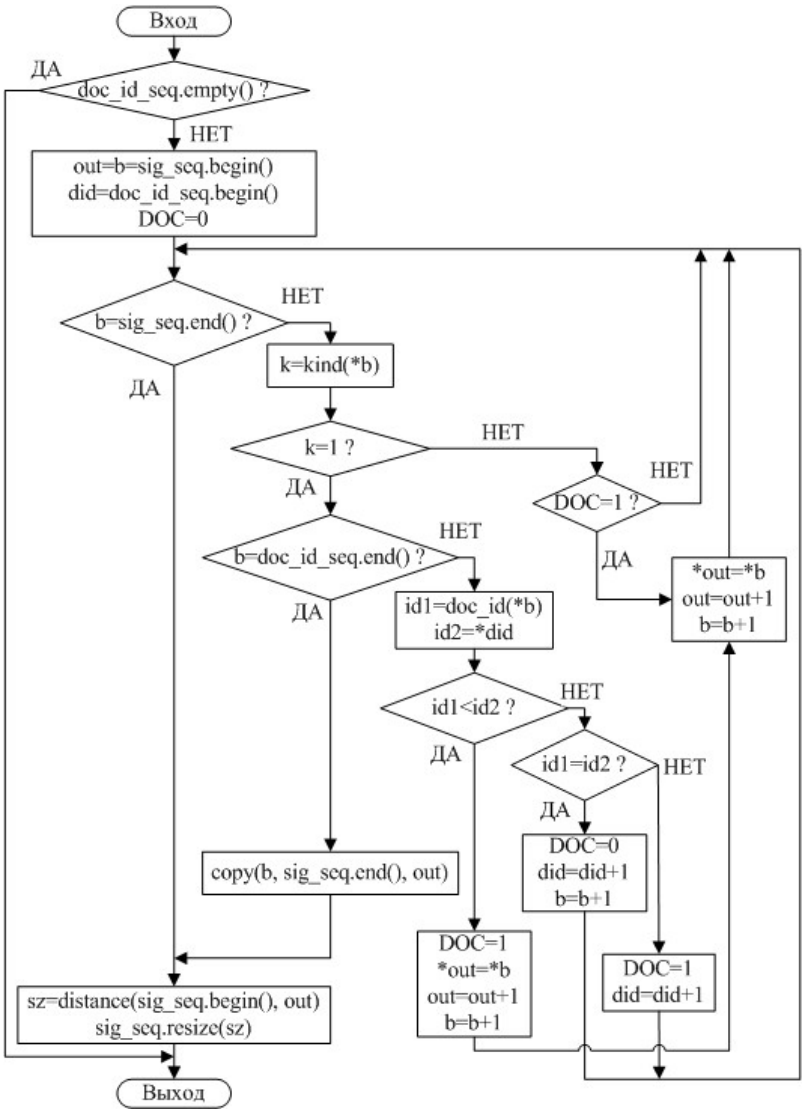

Рис. 8. Блок-схема алгоритма фильтрации σ-последовательности

Отметим, что разработанный подход к наполнению ИПИ позволяет реализовать параллельную обработку ИТД (по вычислительным ядрам на одном вычислительном узле в составе распределённой ИАС).

Заключение

Статья представляет первую часть исследования, посвящённого вопросам реализации сервисов полнотекстового поиска в ИАС. В первой части исследования рассмотрены разработанные структуры данных, предназначенные для реализации поисковых индексов и решения задач полнотекстового поиска. В статье предложен также метод построения поисковых индексов.

ИПИ и метод его построения входят в состав алгоритмического и программного обеспечения ИАС [6]. На основе представленных структур данных реализуются сервисы полнотекстового поиска и тематического анализа коллекций научно-технических документов для отслеживания динамики публикационной активности и анализа качества научно-технических публикаций.

По сравнению с классическими подходами, предложенные структуры данных для пред-

ставления ИПИ требуют меньшего объёма памяти для хранения информации о текстах документов. Кроме того, представленный метод построения ИПИ поддерживает инкрементальное наполнение (обновление и пополнение без необходимости полной перестройки индекса всей коллекции с самого начала) и параллельное индексирование документов.

Практические эксперименты показывают, что индексирование коллекции, содержащей 958 тыс. документов, на одном сервере-индексаторе занимает около 6,5 часов. Общий объём текста на русском языке в коллекции составляет более 8,5 Гб (в кодировке UTF-8). Средняя скорость индексирования составляет около 23 Мб текста в минуту. При этом использована следующая конфигурация средств ИАС:

– количество запущенных потоков лингвистического анализа для построения ИТД – 17 (распределены между серверами ИАС: Intel Core i7 CPU 3.1 GHz, 16 Gb RAM, 4xHDD-500 Gb 7200rpm, RAID 1);

– количество модулей построения и агрегации ИПИ – 1;

– количество потоков агрегации ИПИ – 8.

Во второй части исследования будут рассмотрены алгоритмы оценки релевантности документов и ранжирования результатов поиска, а также вопросы практической реализации предложенных алгоритмов и структур данных в виде сервисов ИАС.

Литература

1. Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.
2. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. — Вильямс, 2011. — ISBN 978-5-8459-1623-5.
3. Осипов Г.С., Смирнов И.В., Тихомиров И.А. Реляционно-ситуационный метод поиска и анализа текстов и его приложения // Искусственный интеллект и принятие решений. М.: ИСА РАН – №2, 2008. С. 3-10.
4. И.В. Соченков. Метод сравнения текстов для решения поисково-аналитических задач // Искусственный интеллект и принятие решений. М.: ИСА РАН, №2, 2013. С.95-106.
5. Смирнов И.В., Соченков И.В., Муравьев В.В., Тихомиров И. А. Результаты и перспективы поискового алгоритма Exactus. // Труды российского семинара по оценке методов информационного поиска РОМИП'2007-2008. Санкт-Петербург: НУ ЦСИ, 2008. - С. 66-76.
6. Osipov, G.; Smirnov, I.; Tikhomirov, I. and Shelmanov, A. Relational-Situational Method for Intelligent Search and Analysis of Scientific Publications. // In Proceedings of the Workshop on Integrating IR technologies for Professional Search Moscow, Russian Federation, March 24, 2013, p.57-64. [Электронный ресурс] URL: http://ceur-ws.org/Vol-968/irps_10.pdf (дата обращения 23.03.2013).
7. D. E. Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Addison-Wesley, (1973), 722 pages.
8. Патент РФ № 2473119, 05.08.2011. Осипов Г.С., Тихомиров И.А., Соченков И.В., Смирнов И.В. Способ и система семантического поиска электронных документов, 2013.
9. Смирнов И.В. Метод автоматического установления значений минимальных синтаксических единиц текста. // Информационные технологии и вычислительные системы. №3, 2008. – С. 30-45.
10. M. Seeger. Key-Value stores: a practical overview. Media, pages 1--21, 2009 // [Электронный ресурс] — URL: http://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf (дата обращения 23.03.2013).
11. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ./ Роберт Сэджвик. – К.: Издательство “ДиаСофт”, 2001. – 688 с.

Соченков Илья Владимирович. Научный сотрудник ООО «Технологии системного анализа», инженер-исследователь ИСА РАН, ассистент кафедры информационных технологий Российского университета дружбы народов. Окончил Российский университет дружбы народов в 2009 году. Автор 25 научных работ. Область научных интересов: интеллектуальные методы поиска и анализа информации, обработка больших массивов данных, защита сетей, контентная фильтрация, компьютерная лингвистика. E-mail: ivsochenkov@gmail.com

Суворов Роман Евгеньевич. Инженер-исследователь ООО «Технологии системного анализа», аспирант ИСА РАН. Окончил Рыбинский государственный авиационный технический университет им. П.А. Соловьева в 2012 году. Автор 5 научных работ. Область научных интересов: интеллектуальный анализ текстовой информации, контентная фильтрация, интеллектуальные динамические системы. E-mail: resuvorov@gmail.com