ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 8
N 11
2017

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.054 DOI: 10.17587/prin.8.483-489

А. О. Коротков, инженер по качеству, e-mail: akorotkov@ec-leasing.ru,
Б. А. Позин, д-р техн. наук, проф., e-mail: bpozin@ec-leasing.ru, ЗАО "ЕС-лизинг", г. Москва

Автоматизация процесса генерации тестовых данных для комплексного функционального тестирования информационных систем, управляемых XML-сообщениями

Описаны результаты реализации инструментальных средств генерации тестовых данных методом разбиения значений входных параметров системы на классы эквивалентности для автоматизации комплексного функционального тестирования информационных систем, управляемых XML-сообщениями.

Ключевые слова: *функциональное тестирование, комплексное тестирование, регрессионное тестирование, автоматизация тестирования, XML-сообщения, классы эквивалентности, генерация XML-сообщений, XSD-схемы, ISO/IEEE/IEC 29119-4*

Введение

В процессе обеспечения жизненного цикла ответственных информационных систем, как правило, выполняют работы по их развитию и доработке. Приоритетной задачей при модификации системы в ходе таких работ является перепроверка уже существующих и тестирование новых функциональных возможностей.

Ключевым положением в проведении такого тестирования является необходимость воспроизведения большого количества ситуаций, чтобы убедиться, что внесенные в ходе модификации изменения не отразились на функциях, которые ранее выполнялись корректно, согласно заявленной документации. Такие проверки требуют большого объема разнотипных данных, и чаще всего эти данные тестировщики формируют вручную. Однако ручное формирование тестовых примеров — весьма трудоемкая задача.

За определенное время накапливается тестовая база, позволяющая запускать проверки в автоматическом режиме. Однако при этом появляются вопросы, связанные с актуализацией данных. В процессе доработок системы может изменяться и ее состояние, и состояние внутренних справочных баз данных, форматы входных/выходных сообщений. Такие изменения могут оказывать прямое влияние и на сами тестовые данные, так как старые тестовые наборы, ранее корректные для системы, могут оказаться недействительными или неверными. Возникает задача актуализации тестовых наборов под новое состояние системы. Чем больше тестовых данных, тем выше трудоемкость по проведению таких работ.

Исходя из изложенного, определим две основные задачи, систематически возникающие при проведении повторного (регрессионного) тестирования:

1) формирование большого объема тестовых данных с высокой степенью покрытия функциональных возможностей системы;

2) актуализация тестовых данных под новые форматы и новые значения в справочных базах системы.

Решение этих задач возможно путем автоматизации процессов генерации тестовых данных. Рассмотрим постановку задачи такой автоматизации.

Постановка задачи автоматизации генерации тестовых данных

Взаимодействие крупных информационных систем в настоящее время, как правило, осуществляется посредством обмена электронными сообщениями. Примером могут послужить web-сервисы [1] или банковские информационные и платежные системы [2] (рис. 1).

При тестировании таких систем имитируют внешние воздействия на целевую систему и отслеживают ее ответную реакцию. Здесь и далее по тексту целевой системой будем называть программную систему, которая выступает в качестве объекта тестирования. Как синоним будем также использовать словосочетание "тестируемая система". В рамках данной статьи под термином "тестирование" понимается проведение комплексных испытаний, по результатам которых можно сделать однозначный вывод о соответствии системы предъявляемым к ней функциональным

Рис. 1. Информационная система, управляемая электронными сообщениями

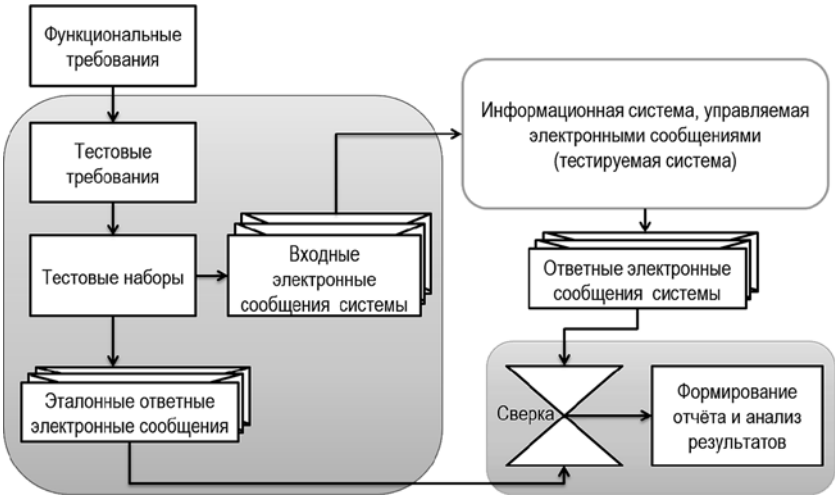

Рис. 2. Процесс тестирования целевой системы

требованиям. На основе функциональных требований формируются требования к характеристикам, составу и последовательности внешних воздействий (в нашем случае — к электронным сообщениям) на тестируемую систему и ее ответной реакции, а также критерии завершения проверки функционального требования. Такие требования будем называть тестовыми. На базе тестовых требований тестировщик создает наборы входных электронных сообщений и, для сверки правильности реакции целевой системы, эталонные (ожидаемые) ответные сообщения, иными словами — наборы тестовых данных (тестовые наборы). Схема проведения комплексного функционального тестирования представлена на рис. 2.

Структура и формат электронных сообщений тестовых наборов могут регламентироваться схемами XML-сообщений — XSD-схемами[1] [3], которые включены в состав самой тестируемой системы. Кроме того, данные, которые содержат в себе электронные сообщения, должны соответствовать бизнес-логике тестируемой системы. При актуализации таких наборов тестовых данных появляется необходимость повторно проверять соответствие каждого типа сообщений схемам и вносить новые значения параметров при изменениях в мастер-данных целевой системы. Выполнение таких действий при ручном формировании тестовых данных влечет за собой большие потери времени и ресурсов.

Автоматизация процесса подготовки данных позволит добиться следующих результатов:

- сократить общее время на проведение тестирования;
- снизить трудоемкость работы тестировщиков;
- увеличить степень покрытия функциональных возможностей системы.

В соответствии со схемой проведения тестирования и требованиями к содержанию электронных сообщений сформулированы перечисленные далее требования к автоматизации процесса проведения комплексного тестирования функциональных свойств системы:

1) генерация тестовых данных должна проходить в автоматизированном режиме;

2) по результатам генерации должен быть сформирован набор тестовых данных, состоящий из входных XML-сообщений и ожидаемых ответных (эталонных) XML-сообщений целевой системы;

3) XML-сообщения должны быть валидными (удовлетворять требованиям, предъявляемым к структуре и содержанию сообщений) в соответствии с XSD-схемами;

4) XML-сообщения должны содержать корректные данные с точки зрения бизнес-логики тестируемой системы;

5) генерируемый набор тестовых данных должен удовлетворять критериям, принятым для проверки функционального требования;

6) сверка ответных сообщений тестируемой системы с эталонными должна проводиться в автоматическом режиме.

Обзор существующих решений

Подавляющее большинство созданных на рассматриваемом направлении инструментальных средств ориентировано на генерацию сообщений с корректной структурой, а заполнение значений параметров предоставляется конечному пользователю. В рамках настоящей статьи не рассматривались веб-сервисы по генерации тестовых данных, так как их возможности значительно меньше в сравнении с Desktop-версиями коммерческих решений. Среди последних следует выделить три наиболее известные и широкие по реализуемым с их помощью функциональным возможностям генератора:

- SoapUI;
- Stylus Studio XML Editor;
- Altova MissionKit.

[1] Помимо схем, определяющих формат XML-сообщений, существуют широко распространенный формат JSON и другие форматы. В рамках данной статьи рассматривается взаимодействие только с XML-сообщениями и схемами.

Таблица 1

Сравнительная характеристика существующих инструментальных средств генерации на их соответствие предъявляемым требованиям

Требование	Продукт		
	SoapUI	Stylus Studio XML Editor	Altova MissionKit
1. Генерация тестовых данных должна проходить в автоматизированном режиме	Выполняется		
2. По результатам генерации должен быть сформирован набор тестовых данных, состоящий из входных XML-сообщений и ожидаемых ответных (эталонных) XML-сообщений целевой системы	Отсутствует автоматизированная генерация ответных сообщений системы и пользователь должен вручную закрепить за каждым тестовым случаем эталонные ответы	Отсутствует инструмент для получения и сверки ответов системы с ожидаемым результатом	Отсутствует возможность генерации связки входных и ответных к ним сообщений
3. XML-сообщения должны быть валидными в соответствии с XSD-схемами	Выполняется		
4. XML-сообщения должны содержать корректные данные с точки зрения бизнес-логики тестируемой системы	Выполняется	Параметры в сообщениях заполняются случайным образом и могут не нести никакой смысловой нагрузки и быть некорректными с точки зрения бизнес-логики тестируемой системы	Выполняется
5. Генерируемый набор тестовых данных должен удовлетворять критериям проверки функционального требования	Отсутствует возможность задавать критерии проверки функционального требования		
6. Сверка ответных сообщений тестируемой системы с эталонными должна проводиться в автоматическом режиме	Выполняется	Отсутствует модуль сверки	

В табл. 1 рассмотрены представленные инструментальные средства по генерации XML-сообщений на базе XSD-схем на соответствие предъявляемым требованиям.

Анализ рынка показывает, что на настоящее время не существует продукта, позволяющего с его помощью автоматизировать проведение комплексного функционального тестирования систем, управляемых электронными сообщениями, в соответствии с заданными требованиями. С учетом изложенного выше задача автоматизации процесса генерации тестовых данных для комплексного функционального тестирования информационных систем, управляемых XML-сообщениями, является актуальной.

Решения по формализации описания задания на генерацию тестовых данных

Для описания правил формирования набора тестовых данных в автоматизированном режиме принято решение разработать декларативный язык LDL (*logic data language*). Данный язык позволяет описать тестовые требования в понятном для машины виде. В состав языка входят конструкции, позволяющие определять структуры входного и ответных к нему сообщений в соответствии с их XSD-схемами. Для наполнения этих элементов данными предусмотрены несколько видов источников:

- введенные пользователем значения — какие-либо константные значения, заданные тестировщиком вручную, процедуры преобразования данных либо правила проведения генерации значений определенного типа;
- мастер-данные — корректные с точки зрения бизнес-логики тестируемой системы данные, которые как правило, содержатся в самой системе и могут изменяться с течением времени или в зависимости от проверок, поэтому в языке предусмотрена конструкция для описания правил получения таких данных непосредственно из целевой системы;
- накопленные данные — конструкция, позволяющая описывать возможные в ходе генерации ситуации, при которых необходимо провести промежуточные расчеты и/или сохранить полученное значение с последующим использованием при формировании ответных сообщений.

Тестовые требования, помимо требований к содержимому набора тестовых данных, определяют также и критерии проверки функционального требования. В международном стандарте [4] описывается метод формирования тестовых данных с помощью разбиения входных значений по классам эквивалентности [5], который обеспечивает высокую степень покрытия при должной квалификации тестировщика. Таким образом, реализованный механизм описания классов эквивалентности в языке LDL предоставляет тестировщику возможность задавать для каждого элемента сообщения только те значения, которые необходимы для проверки функционального требования. Однако при описании классов эквивалентности для нескольких элементов сообщения возникает проблема их сочетания — при переборе всех возможных комбинаций классов эквивалентности между собой могут встретиться сочетания, которые не происходят в реальной жизни и/или некорректны с точки зрения бизнес-логики целевой системы и не имеют смысла. Во избежание генерации таких данных в языке реализованы механизмы для описания недопустимых комбинаций классов эквивалентности и различного рода ограничения на них. Структурная схема языка LDL представлена на рис. 3.

Решение по автоматизации генерации тестовых данных

Для автоматизации процесса генерации тестовых данных для комплексного функционального тестирования информационных систем, управляемых XML-сообщениями, по заданиям на языке LDL разработан инструментарий генерации, состоящий из двух модулей:

- генератор тестовых данных по классам эквивалентности, который считывает правила проведения генерации и на их основе формирует наборы тестовых данных;
- модуль сверки результатов, который проводит сверку ответов тестируемой системы с ожидаемыми результатами и, по итогам работы, формирует отчет, содержащий информацию о каждом сообщении, поданном на вход целевой системы, полученных на него ответов от системы и результаты сверки этих ответов с эталонными результатами.

Для проведения функционального тестирования с применением генератора тестовых данных по классам эквивалентности (рис. 4) необходимо провести предварительную настройку генератора, в ходе ко-

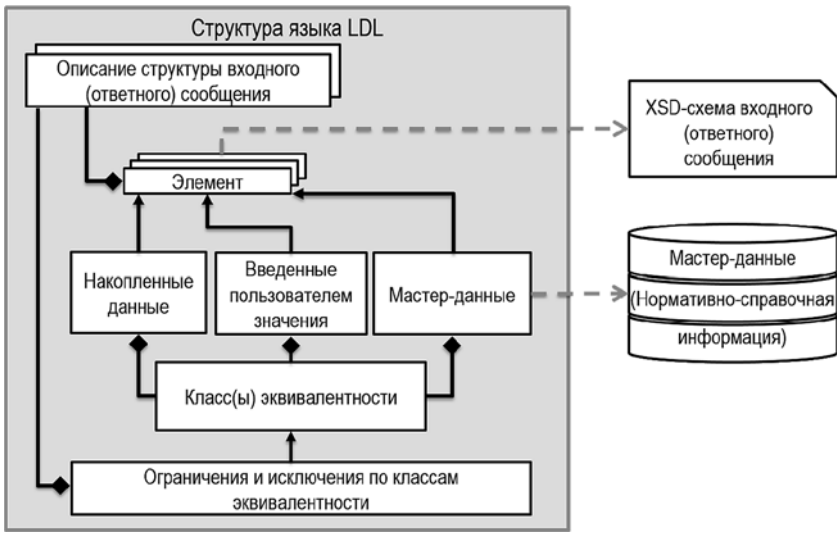

Рис. 3. Структурная схема языка LDL

Рис. 4. Проведение тестирования с применением инструмента генерации тестовых данных

торой устанавливаются соединения с базами данных тестируемой системы, определяются ссылки на форматы сообщений системы, а также происходит настройка модуля сверки, при которой указываются точки тестируемой системы, в которых необходимо собирать ответные сообщения.

На вход генератора подается сформированное тестировщиком на языке LDL описание правил проведения генерации, запускается генератор. По окончании работы генератора будут сформированы наборы входных электронных сообщений, которые на следующем этапе подаются в тестируемую систему, и эталонных ответов к ним, которые загружаются в специально разработанный модуль сверки результатов. Сверка происходит в потоковом автоматизированном режиме: как только в модуль сверки поступает результат работы системы, он сразу проходит проверку на соответствие эталону. По окончании работы модуль сверки автоматически формирует отчет, в котором указана информация об успешном или неудачном прохождении тестов.

Результаты практического внедрения

Представленное выше инструментальное средство генерации тестовых данных внедрено в одном из российских банков. Время его освоения персоналом в среднем составило около трех недель. Однако уже по итогам первого месяца наблюдался значительный прирост базы накопленных тестов. Необходимо также отметить, что снизился риск проявления ошибки в тестовых данных. Причина в том, что большинство значений вводились не тестировщиком, как в случае с ручным тестированием, а выбирались из базы данных самой информационной системы или генерировались в заданных пользователем генератора пределах.

В табл. 2 приведена сравнительная характеристика тестовых случаев (шесть различных по воздействию на тестируемую систему типов входных XML-сообщений), накопленных в ходе ручного тестирования за 5 лет, и тестовых случаев, полученных при применении инструментального средства генерации тестовых данных.

Из данных табл. 2 можно сделать вывод, что степень покрытия системы при применении рассматриваемого средства генерации тестовых данных возросла более чем в 6 раз. Однако здесь необходимо отметить, что нет точной информации, какое именно функциональное свойство было протестировано с помощью описанного инструментария генерации тестовых данных. Следовательно, потенциальная степень тестового покрытия при его применении может быть как больше, так и меньше, в зависимости от квалификации тестировщика.

До внедрения инструментального средства генерации тестовых данных тестировщику необходимо было просматривать все ответы системы вручную, чтобы выявить наличие и характер несоответствий результатов работы системы ожидаемым результатам. Теперь, несмотря на то, что тестовая база значительно увеличилась, при использовании модуля сверки существенно сократились затраты времени на поиск и выявление неуспешно пройденных тестов. Причина в том, что все ответы системы проходят сравнение с эталонами, полученными с помощью средства генерации тестовых данных, в автоматическом режиме. Таким образом, тестировщик получает подробный отчет о тестовых случаях, которые прош-

Таблица 2

Сравнительная характеристика применения инструментального средства генерации тестовых данных и накопленной базы ручного тестирования

Тип сообщений тестируемой системы	Накопленная за пять лет ручной работы база тестировщиков		При использовании инструмента генерации тестовых данных в течение трех месяцев	
	Число входных сообщений	Число ответных сообщений	Число входных сообщений	Число ответных сообщений
Тип 1	125	312	1351	3378
Тип 2	657	2190	4328	14431
Тип 3	398	3551	2122	18903
Тип 4	1741	5946	10354	35198
Тип 5	774	3864	5175	34279
Тип 6	2033	4992	12179	29877
Итого	5728	20 855	35509	136066

Таблица 3

Трудоемкость анализа результатов

Характеристика	При ручном тестировании	При применении инструментального средства генерации тестовых данных
Число входных сообщений	5728	35 509
Анализируемые тестовые ситуации	Все тестовые случаи	Только неудачно проведенные тесты (в среднем 0,8...1%)
Итого число проверяемых тестовых ситуаций	5728	330...350

ли неуспешно, и может сразу приступить к анализу данных ситуации и регистрации инцидентов, если выявится дефект системы. В табл. 3 приведены сведения о трудоемкости анализа результатов при проведении ручного тестирования и при применении инструментального средства генерации тестовых данных совместно с модулем сверки.

На основе показателей в табл. 3 можно сделать вывод о том, что трудоемкость анализа результатов сократилась более чем в 10 раз. Однако следует уточнить, что время, затрачиваемое тестировщиком на проверку корректно отработавших тестов, по сравнению со временем на проведение анализа неудачно завершившихся тестов значительно меньше. Среднее время, которое тратит тестировщик на анализ корректно выполненного теста, составляет 2 мин, тогда как анализ и, при необходимости, регистрация инцидента по неудачно пройденному тесту в среднем составляет 20 мин.

Исходя из этих данных, вычислим трудоемкость анализа результатов для случая проведения ручного тестирования (р.т.), а также для случая проведения тестирования с помощью инструментального средства генерации тестовых данных (а.т.), используя для этого следующую формулу:

$$\text{трудоемкость тестирования}\,(\text{ч.часов}) = \frac{N_{\text{вх.сооб}} \cdot \%_{\text{неуд}} \cdot t_{\text{неуд}} + N_{\text{вх.сооб}} \cdot \%_{\text{успешных}} \cdot t_{\text{усп}}}{60 \text{ мин}}, \quad (1)$$

где $N_{\text{вх.сооб}}$ — число входных сообщений; $\%_{\text{неуд}}$ — % неудачно завершившихся тестов; $t_{\text{неуд}}$ — время, затрачиваемое тестировщиком на обработку неудачно проведенных тестов и регистрацию инцидента; $\%_{\text{успешных}}$ — % успешно завершившихся тестов; $t_{\text{усп}}$ — время, затрачиваемое тестировщиком на обработку успешно проведенных тестов.

В результате вычислений по формуле (1) получаем:

$$\text{трудоемкость}_{\text{р.т}} = \frac{5728 \cdot 0,01 \cdot 20 + 5728 \cdot 0,99 \cdot 2}{60} \approx 208 \text{ человеко-часов};$$

$$\text{трудоемкость}_{\text{а.т}} = \frac{35\,509 \cdot 0,01 \cdot 20 + 35\,509 \cdot 0,99 \cdot 0}{60} \approx 118 \text{ человеко-часов};$$

$$\frac{\text{Трудоемкость}_{\text{р.т.}}}{\text{Трудоемкость}_{\text{а.т.}}} \approx 1,76.$$

Как следует из представленных вычислений, трудозатраты на анализ результатов сократились приблизительно в 1,76 раз, однако количество проверок системы было увеличено более чем в 6 раз.

Заключение

С учетом представленных выше результатов практической апробации можно сделать следующие выводы.

Инструментальное средство генерации тестовых данных может найти применение для разных предметных областей. Его можно настроить под любую архитектуру системы, обрабатывающую XML-сообщения и имеющую описание видов этих сообщений в формате XSD-схем. Разработанный язык LDL позволяет решить задачу актуализации данных под новые значения в базах данных тестируемой системы. При внесении изменений в XSD-схемы необходимо изменить лишь LDL-файл и повторить генерацию, что значительно снижает трудоемкость поддержания актуальной структуры XML-сообщений в наборах тестовых данных.

В планах развития инструментального средства генерации тестовых данных и языка LDL предполагается добавление новых форматов генерации, в первую очередь — JSON, а также расширение возможных источников данных.

Список литературы

1. **Яковенко П. Н., Сапожников А. В.** Инфраструктура тестирования веб-сервисов на базе технологии TTCN-3 и платформы .Net // Труды Института системного программирования РАН. 2009. Т. 17. С. 63—74.

2. **Унифицированные** форматы электронных банковских сообщений Банка России. Обмен с клиентами Банка России. URL: http://cbr.ru/analytics/?PrtId = Formats (дата обращения: 10.07.2017).

3. **W3C** XML Schema Definition Language (XSD). URL: https://www.w3.org/TR/xmlschema11-1/ (дата обращения 21.06.2017).

4. **Software** and systems engineering — Software Testing — Part 4: Test Techniques. ISO/IEC/IEEE 29119-4 — 2015. P. 7—8.

5. **Майерс Г., Баджетт Т., Сандлер К.** Искусство тестирования программ 3-е изд. М.: Вильямс, 2012. 272 с.

Automation of the Test Data Generation Process for Complex Functional Testing for XML Message Driven Systems

A. O. Korotkov, akorotkov@ec-leasing.ru, **B. A. Pozin**, bpozin@ec-leasing.ru, EC-leasing, Moscow, 117405, Russian Federation

Corresponding author:

Korotkov Aleksandr O., Quality Engineer, EC-leasing, Moscow, 117405, Russian Federation, E-mail: akorotkov@ec-leasing.ru

Received on August 21, 2017
Accepted on September 04, 2017

The article considers the problem of conducting functional regression testing of XML message driven systems and its solution by automating the process of preparing test cases and analyzing the results of testing.

The main goals of this research:

- *reduction of the time spent on the creation and maintenance of test cases;*
- *increase testers efficiency;*
- *increase in the total number of test cases;*
- *increase test coverage;*
- *reduce the time spent on the analyzing the results of testing.*

The article describes the results of development of the tool for generating XML-messages and its applying to automate complex functional testing. Implements the testing method described in international standard ISO/IEC/IEEE 29119-4:2015 testing method and based on equivalence partitioning for XML messages driven systems. Built-in declarative language gives the test developer ability to describe equivalence partitions for XML-messages with different structure and different content. The language allows describing data sources for elements and attributes of XML messages. For each element, both limits and dependences can be set. The tool allows one to generate XML messages based on XSD schemas using elements values that matched the equivalence partitions description from selected data sources. It gives ability to create numerous different test cases for current state of normative-reference data in testing system. The developed tool can be widely used for testing XML message driven systems.

Keywords: *automated testing, functional testing, regression testing, XML messages, XML messages generation, equivalence partitions, XML messages driven systems, ISO/IEEE/IEC 29119-4*

For citation:

Korotkov A. O., Pozin B. A. Automation of the Test Data Generation Process for Complex Functional Testing for XML Message Driven Systems, *Programmnaya Ingeneria*, 2017, vol. 8, no. 11, pp. 483—489.

DOI: 10.17587/prin.8.483-489

References

1. **Jakovenko P. N., Sapozhnikov A. V.** Infrastruktura testirovanija veb-servisov na baze tehnologii TTCN-3 i platformy .Net (Infrastructure for testing Web services based on TTCN-3 technology and .Net platform), *Trudy Instituta sistemnogo programmirovanija RAN*, 2009, vol. 17, pp. 63—74 (in Russian).

2. **Unificirovannye** formaty jelektronnyh bankovskih soobshhenij Banka Rossii. Obmen s klientami Banka Rossii (Unified formats for electronic banking messages of the Bank of Russia. Exchange with customers of the Bank of Russia), available at: http://cbr.ru/analytics/?PrtId=Formats

3. **W3C** XML Schema Definition Language (XSD), available at: https://www.w3.org/TR/xmlschema11-1/

4. **Software** and systems engineering — Software Testing — Part 4: Test Techniques, ISO/IEC/IEEE 29119-4 — 2015, pp. 7—8.

5. **Myers G. J., Sandler C., Badgett T.** *Iskusstvo testirovanija program* (The Art of Software Testing), 3rd Ed., Moscow, Williams, 2012, 272 p. (in Russian).

УДК 004.624 DOI: 10.17587/prin.8.490-503

В. А. Васенин, д-р физ.-мат. наук, проф., e-mail: vasenin@msu.ru, НИИ Механики МГУ имени М. В. Ломоносова, **М. Д. Дзабраев,** разработчик, e-mail: dzabraew@gmail.com, ИАС ИСТИНА, г. Москва

Методы и средства мониторинга публикаций в средствах массовой информации

Описан подход к решению востребованной на практике задачи извлечения данных с веб-сайтов в целях их дальнейшей обработки для тех или иных приложений. Изложено описание и детали реализации алгоритма, с помощью которого представляется возможным осуществить обход веб-страницы. Целью обхода веб-страницы является попадание во все возможные места на веб-странице и извлечение полезных данных. Обход веб-страницы осуществляется путем нажатия кнопок на веб-странице. Нажатие каждой кнопки способно либо загрузить новую веб-страницу, либо модифицировать существующую с помощью исполнения JavaScript. Алгоритм, описанный в настоящей статье, предназначен для реализации нажатия кнопок, которые изменяют текущую веб-страницу.

Ключевые слова: *извлечение данных, веб, readability, обход веб-сайта, обход веб-страницы, Javascript, Firefox*

Введение

Средства массовой информации с использованием современных инфокоммуникационных технологий перманентно генерируют, аккумулируют и хранят огромные объемы сведений, имеющих отношение к тем или иным субъектам политической и экономической, научной-технической и хозяйственной, социальной и других сфер деятельности. Эти сведения способствуют формированию отношения отдельных людей, а также больших социальных групп к этим субъектам, влияют на их имидж в обществе. Как следствие, сформированное отношение оказывает прямое или опосредованное воздействие на результаты деятельности перечисленных субъектов. С учетом отмеченных обстоятельств задача создания эффективных средств автоматизации процессов выявления текстов в средствах массовой информации (далее СМИ), имеющих отношение к отдельным субъектам перечисленных выше сфер деятельности в обществе, приобретает особую актуальность.

В работе [1] представлены подходы к решению этой задачи, макет программных средств ее решения и результаты анализа тональности публикаций в СМИ с использованием таких средств. Одним из существенных недостатков этого макета является отсутствие в нем механизмов эффективного перманентного мониторинга заранее определенного и достаточно широкого информационного пространства (множества веб-сайтов) для формирования исходной коллекции документов в целях ее последующего анализа. Это обстоятельство значительно сужало функциональные возможности представленного в работе [1] макета для комплексного решения поставленной выше задачи.

В настоящей статье изложены модельные представления, алгоритмы и реализующие их программные механизмы, которые выполняют перманентный мониторинг заданного множества веб-сайтов в целях обнаружения и извлечения содержащихся в них новых публикаций. Извлеченные публикации при этом сохраняются в специально созданную базу данных (далее БД) для последующего их анализа с той или иной целью. Совокупность программ, которые реализуют перечисленные выше функциональные возможности, далее будем именовать сервисом.

На настоящее время сервис в автоматическом режиме осуществляет извлечение публикаций с четырех веб-сайтов новостных агентства: ria.ru, lenta.ru, gazeta.ru, mk.ru. Количество новостных сообщений (публикаций), хранящихся на данный момент в БД, насчитывает 3 млн 200 тыс. Средствами этого сервиса осуществляется мониторинг публикаций на предмет содержания одного из заранее заданных названий организации с последующей классификацией публикаций на три класса тональности.

В следующем разделе представлено описание модели, на основе которой построен алгоритм, положенный в основу сервиса, осуществляющего извлечение публикаций из веб-сайтов.

Автоматическое извлечение данных из веб-сайтов

Веб-сайт представляет собой набор документов. Каждый документ обладает уникальным адресом — URL. Как правило, документ — это смесь текста на естественном языке и HTML-разметки. Практически всегда вместе с документом распространяются стили (CSS), с помощью которых определяется вид документа и программы на языке программирования JavaScript. Стили и JavaScript-программы создают разработчики веб-сайта.

С помощью JavaScript-кода страницы веб-сайта могут быть наделены динамическим (интерактивным) поведением. Примером такого поведения является отслеживание событий, которые порождаются пользователем. Можно запрограммировать реакцию веб-страницы на то или иное событие, которое является результатом действий пользователя. С помощью JavaScript-кода с содержанием HTML-документа можно сделать многое, вплоть до полного его удаления. Пример изменения документа с помощью JavaScript-кода представлен на рис. 1 (см. вторую сторону обложки). На данном рисунке выделена кнопка, при каждом нажатии которой в документ (веб-страницу) встраивается список ссылок на новостные сообщения.

Для извлечения данных с веб-сайта необходимо собрать коллекцию URL, которые указывают на обрабатываемый веб-сайт. При этом желательно суметь собрать как можно больше URL. Причина в том, что если какой-то URL оказывается пропущенным, то вместе с ним упущенным становится и документ, который соответствует данному URL.

Поскольку чрезвычайно важным является сбор как можно большего числа URL, то это вынуждает разработчика писать программы, которые не просто загружают HTML-документ по данному URL и извлекают из него все URL, но также поддерживают динамический режим взаимодействия с веб-сайтом, "нажимая" кнопки и генерируя различные события. Иными словами, программа, которая проводит сбор URL, должна уметь исполнять JavaScript-код. Причина в том, что каждое нажатие той или иной кнопки может встроить в документ набор URL, которые ранее не были обнаружены.

Например, если нажимать отмеченную на рис. 1 (см. вторую сторону обложки) кнопку, то последовательно в документ будут встроены все ссылки на все публикации для данного агентства. Если программа будет обеспечивать такое нажатие на указанную кнопку, то она гарантированно извлечет URL всех публикаций.

Можно попытаться не исполнять JavaScript-код, а поступить напрямую: загружать документ, извлекать из него все URL; добавлять каждый URL в БД; аналогично поступать для каждого URL, который хранится в БД. Однако, исполняя JavaScript-код, программа сможет собрать не меньшее количество URL, чем без исполнения JavaScript-кода.

Обход веб-страницы с исполнением JavaScript-кода

Рассмотрим неформальное описание модели, которая положена в основу алгоритма, реализующего автоматический обход веб-страницы. Этот алгоритм предписывает, как нужно обеспечивать нажатие кнопок на веб-странице, чтобы программа смогла извлечь наибольшее число URL. Таким образом, алгоритм преследует следующую цель: в результате нажатия кнопок программа должна попасть во все части веб-страницы, куда такая возможность предоставляется и, оказавшись в каждой из этих частей, извлечь интересующие данные. Алгоритм необходим, чтобы полностью или частично снять с программиста работу по написанию кода для осуществления нажатий кнопок. Перед формализованным изложением алгоритма рассмотрим пример, с помощью которого на вербальном уровне излагаются исходные посылки модели, используемой для его построения, на основе которой разработан этот алгоритм.

На рис. 2 (см. вторую сторону обложки) представлен пример двух кнопок, которые обведены красными овалами. При нажатии кнопки "выбрать по дате" исполняется JavaScript-код, который встраивает в документ прямоугольник "Архив новостей". Этот прямоугольник содержит кнопку "стрелка влево" и набор гиперссылок (URL) в виде календаря. При нажатии кнопки "стрелка влево" будет выполнен JavaScript-код, который обновит содержимое календаря путем удаления имеющихся гиперссылок и добавлением новых, имеющих отношение к предыдущему месяцу гиперссылок. При этом, если нажать "выбрать по дате", затем несколько раз нажимать "стрелка влево", а после этого нажать в любую точку документа, которая не принадлежит прямоугольнику "Архив новостей", то выполнится JavaScript-код, который удалит прямоугольник "Архив новостей". Вместе с прямоугольником будет удалена кнопка "стрелка влево". Возможна ситуация, когда кнопку "стрелка влево" можно было бы нажать еще раз и получить URL, который ранее не встречался. В связи с этим обстоятельством необходимо, чтобы программа запомнила то место, где была ненажатая кнопка "стрелка влево", и запомнила путь, по которому она попала в это место. В дальнейшем программа должна вновь пройти по пути, который был запомнен ранее, и нажать еще не нажатую кнопку.

Перед формальным описанием алгоритма следует подробнее остановиться на том, как устроен HTML-документ и каким образом разработчики веб-сайта встраивают JavaScript-код в такой документ.

Всякий HTML-документ представляет собой дерево HTML-тегов. Дерево HTML-тегов принято называть DOM-дерево. Пример DOM-дерева изображен на рис. 3.

Для понимания принципов работы алгоритма, который будет представлен далее, важно понимать, что такое кнопка в документе с точки зрения DOM-дерева. Кнопка, на которую пользователь мо-

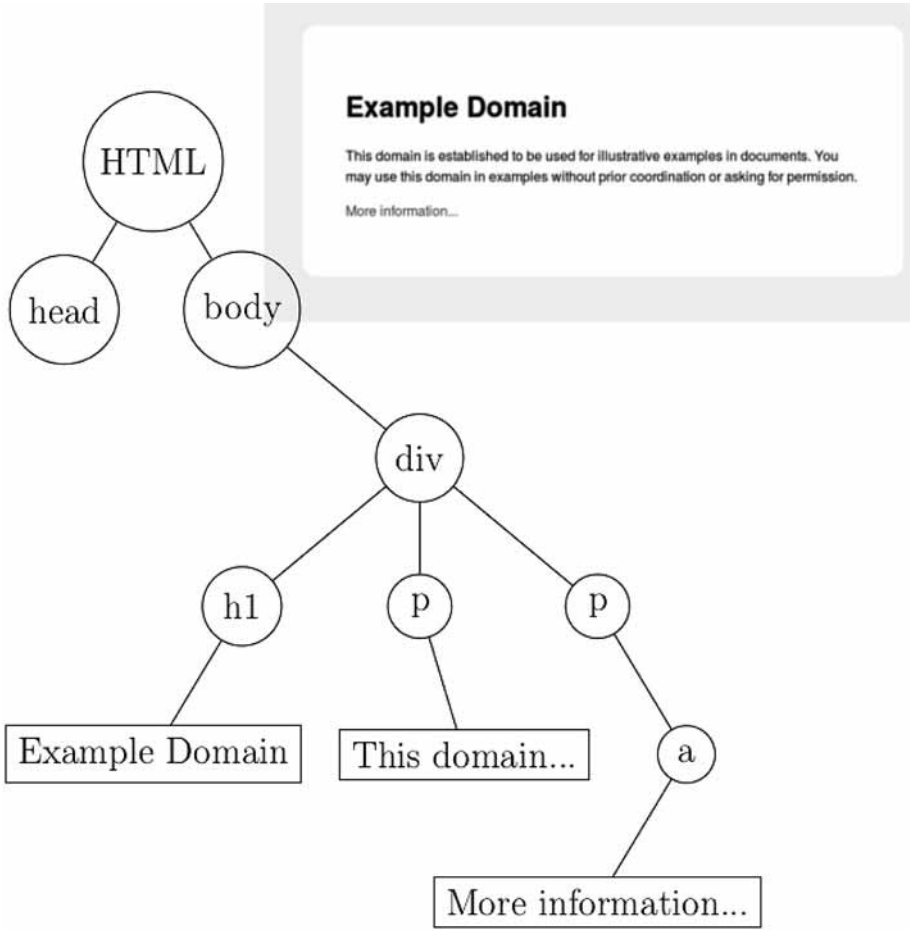

Рис. 3. Фрагмент веб-сайта example.com, который можно наблюдать через веб-браузер, и дерево HTML-тегов, соответствующее фрагменту

жет нажать, является узлом DOM-дерева. При этом отметим, что на этот узел разработчики веб-сайта прикрепили один или более обработчиков событий. Обработчик события представляет собой следующую пару: функция на языке JavaScript; тип события. Если на узел DOM-дерева было сгенерировано событие некоторого типа (обозначим название типа символом T), то будут выбраны все пары, у которых второй компонент совпадает с T, и будут исполнены все функции из первого компонента пары. В рамках алгоритма будет рассматриваться только событие click — нажатие левой кнопки мыши. Таким образом, кнопкой называется узел DOM-дерева, на котором существует хотя бы один обработчик события click.

Для работы алгоритма необходимо отличать одну кнопку от другой. В основу такого различия можно было бы положить следующий способ: каждой кнопке сопоставить путь в DOM-дереве от корня; если пути различаются, то и кнопки можно считать отличающимися. Вместе с тем такой способ не подходит, поскольку при нажатии какой-либо кнопки местоположение нажатой или другой кнопки в дереве может измениться. При повторной загрузке с сервера документа, он также может выглядеть немного иначе с точки зрения древовидной структуры. При этом одни и те же кнопки могут иметь различные пути от корня.

Наиболее корректным представляется следующее отношение равенства, которое неформально можно определить следующим образом: если две кнопки при нажатии на выходе реализуют один и тот же результат, то это одна и та же кнопка. Практическая реализация этого критерия равенства представляет большую сложность. Дело в том, что функции-обработчики, скорее всего, будут использовать на чтение и запись глобальные переменные, а также будут читать/изменять атрибуты DOM-дерева. Ввиду большой сложности программной реализации данного критерия равенства, в предлагаемом подходе этот критерий использоваться не будет.

Введем следующее отношение равенства, которое близко к отношению "делают одно и то же". Каждой кнопке ставятся в соответствие два параметра: текст внутри кнопки; список исходных текстов на JavaScript обработчиков событий. Отношение равенства выглядит следующим образом. Если у двух кнопок указанные параметры совпадают, то это одна и та же кнопка.

Для изложения алгоритма необходимо ввести определение состояния веб-страницы. После загрузки веб-страница имеет некоторое начальное состояние. После нажатия кнопки может ничего не происходить и, таким образом, страница остается в неизменном состоянии. Если после нажатия что-то изменилось, то веб-страница находится в другом состоянии. Например, на рис. 2 (см. вторую сторону обложки) можно выделить следующую цепочку состояний.

1. Состояние после загрузки страницы. Ни одна кнопка не была нажата.

2. После нажатия кнопки "выбрать по дате" в страницу встраивается прямоугольник "Архив новостей". Этот прямоугольник помимо прочего привнес новую кнопку и набор URL. При этом заметим, что встроенных URL до нажатия кнопки на странице не было.

3. После нажатия кнопки "стрелка влево" происходит обновление календаря в прямоугольнике. В процессе обновления календаря происходит удаление имеющихся URL и встраивание новых URL. В результате выполненных изменений состояние веб-страницы изменилось.

4. Если теперь нажать мышью вне прямоугольника, то прямоугольник "Архив новостей" пропадет,

и веб-страница окажется в исходном состоянии (как после загрузки страницы).

На веб-страницу можно смотреть, как на (конечный) автомат. В качестве объектов, которые подаются на его вход, можно рассматривать события нажатия кнопок. При получении объекта на вход состояние автомата меняется.

Выходом автомата является та веб-страница, которую видит пользователь.

Поскольку цель обхода веб-страницы заключается в извлечении URL, то целесообразно определять контекст страницы по набору URL и по набору кнопок. Причина в том, что нажатие любой кнопки может породить операцию встраивания новых URL в страницу. Таким образом, введем следующее определение.

Состоянием веб-страницы называется тройка (B, N, H), где $B = \{b_1,..., b_n\}$ представляет множество кнопок; N — число URL, которые встречаются в данном состоянии страницы. Пусть $h_1,..., h_N$ — множество всех URL на странице. Пусть f (*str*) — хэш-функция (на выходе целое число). Тогда третий параметр определяется соотношением $H = \sum_{i=1}^{N} f(h_i)$.

Изложим далее алгоритм обхода веб-страницы. В процессе обхода веб-страницы будет осуществляться построение графа состояний. Граф, который будет построен в результате обхода веб-страницы, выступает в роли хранителя кнопок. Если в результате нажатия кнопки веб-страница перешла в новое состояние, а в старом состоянии остались ненажатые кнопки, то необходимо запомнить информацию о ранее ненажатых кнопках, а также путь, по которому алгоритм пришел в вершину (состояние), чтобы вновь прийти в это состояние и нажать ненажатые кнопки. В графе каждая вершина будет соответствовать состоянию веб-страницы. Каждой вершине приписывается множество кнопок. Вершины соединяются ориентированными ребрами. Каждое ребро соответствует переходу из одного состояния в другое посредством нажатия кнопки. Каждому ребру при этом приписывается кнопка, с помощью которой был осуществлен переход.

Далее будет представлено формальное описание алгоритма, построенного на базе неформально представленной модели.

Формальное описание алгоритма

Шаг 1. Осуществляется загрузка страницы. После загрузки веб-страницы вычисляются параметры (B_1, N_1, H_1). Затем в граф добавляется корневая вершина. Корневой вершине приписываются вычисленные параметры. Корневой вершине назначается статус текущей вершины. Корневая вершина изображена на рис. 4.

Шаг 2. Если множество B_1 пусто, то обход закончен. Иначе выбирается кнопка $b \in B_1$, после чего она нажимается.

Шаг 3. После того, как нажатие кнопки завершилось, вычисляются параметры (B_2, N_2, H_2).

Шаг 4. Если (B_2, N_2, H_2) $\neq$ (B_1, N_1, H_1), то в граф состояний добавляется новая вершина. Новая вершина назначается текущей вершиной (рис. 5).

В противном случае реализуется петля (рис. 6).

Каждому ребру приписывается список пар (*b*, *hist*): кнопка; история нажатий. Первый элемент пары представляет кнопку, при нажатии которой был осуществлен переход по ребру. Второй параметр пары — история нажатий — является второстепенным, по-

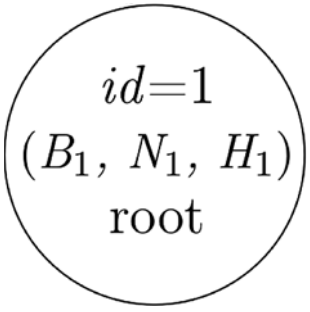

Рис. 4. Корневая вершина в графе состояний

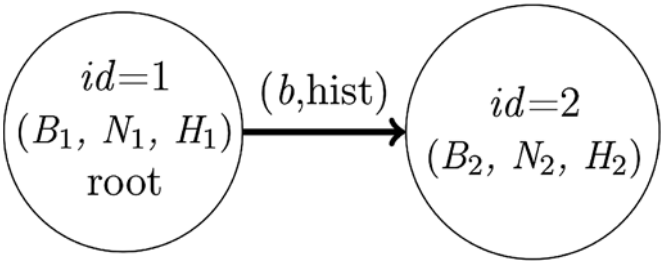

Рис. 5. Добавление в граф новой вершины

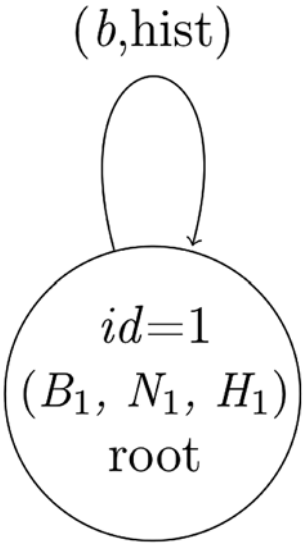

Рис. 6. Петля в графе состояний

этому не будет обсуждаться в настоящем разделе, ему посвящен раздел "История нажатий".

Случай добавления ребра и вершины для первого нажатия ничем не отличается от добавлений вершин и ребер при последующих нажатиях. В общем случае правило добавления вершины и ребра в граф состояний выглядит следующим образом. После нажатия кнопки вычисляются параметры (B, N, H). Далее в графе осуществляется поиск вершины с параметрами (B, N, H). Результату поиска будет соответствовать один из следующих трех пунктов.

- В графе не оказалось вершины с параметрами (B, N, H). В этом случае в граф добавляется вершина (обозначена $id = p$ на рис. 7, *а*) с параметрами (B, N, H). Текущая вершина соединяется ребром с добавленной вершиной. Ребру приписывается нажатая кнопка.
- В графе нашлась вершина A (обозначена $id = p$ на рис. 7, *б*) с параметрами (B, N, H), но не существует ребра из текущей вершины в вершину A. В этом случае добавляется ребро из текущей вершины в вершину A. Ребру приписывается нажатая кнопка.
- В графе нашлась вершина A (обозначена $id = p$ на рис. 7, *в*) с параметрами (B, N, H), причем существует ребро из текущей вершины в вершину A. В этом случае, если ребру уже приписана кнопка, которая была нажата, то никаких действий не предпринимается. Если данному ребру не приписана нажатая кнопка, то кнопка приписывается ребру.

Шаг 5. Для дальнейшего описания алгоритма необходимо дать более подробное описание операции переходов по графу. Для наглядности операция переходов проиллюстрирована на примере рис. 8 (см. третью сторону обложки). Текущая вершина обозначена синим цветом. Следует отметить, что при операциях нажатий кнопок для наглядности имеет смысл представлять веб-браузер с загруженной страницей. Затем необходимо представить, что на странице есть кнопка, а после нажатия этой кнопки страница изменилась, т. е. перешла в новое состояние. Новому состоянию будет соответствовать новая вершина графа и новое ребро, соединяющее старое состояние с новым. Представление о веб-браузере оправдано, поскольку реализация алгоритма основана на веб-браузере Firefox. Если алгоритму требуется перейти из вершины $id = 2$ в вершину $id = 1$, то сначала будет осуществлен переход по ребру (2, 3) посредством нажатия кнопки b_2. Затем будет осуществлен переход по ребру (3, 1), посредством нажатия кнопки b_3. Находясь в вершине $id = 1$, алгоритм может перейти вновь в вершину $id = 2$ посредством нажатия кнопки b_1 или b_4.

Особым случаем операции перехода является переход в корневую вершину графа с помощью перезагрузки страницы. Находясь в любой вершине, можно сделать перезагрузку (обновление) страницы в веб-браузере. В результате перезагрузки страницы алгоритм окажется в начальном состоянии. Следует отметить, что после обновления страницы и вычисления параметров (B, N, H), данные параметры могут не совпадать с параметрами, которые записаны в корневой вершине. Описанная ситуация будет обсуждена в разделе "История нажатий".

Шаг 6. Выбор кнопки для нажатия. Согласно алгоритму, выбор кнопки для нажатия осуществляется с помощью схемы, изображенной на рис. 9. Пусть в результате работы алгоритма был построен некоторый граф, и вершина V этого графа является текущей. Пусть B' представляет множество ненажатых кнопок в данной вершине. Рассмотрим самый верхний блок на рис. 9 и будем двигаться от него вниз. Если в текущей вершине множество B' не пусто, тогда из B' выбирается произвольная кнопка (переход по левой стрелке). Если в текущей вершине множество B' пусто (переход по правой стрелке), то осуществляется поиск пути из текущей вершины до любой другой, в которой B' не пусто. Если такой путь существует, то осуществляется переход по этому пути и выбирается кнопка из множества ненажатых кнопок новой вершины. Если из текущей вершины не существует пути до вершины, в которой есть хотя бы одна ненажатая кнопка, и текущая вершина является корневой вершиной, то процесс обхода считается завершенным. Если текущая вершина не является корневой, то осуществляется переход в корневую вершину и операции, изложенные в данном пункте, выполняются для корневой вершины.

Схематично работа алгоритма изображена на рис. 10.

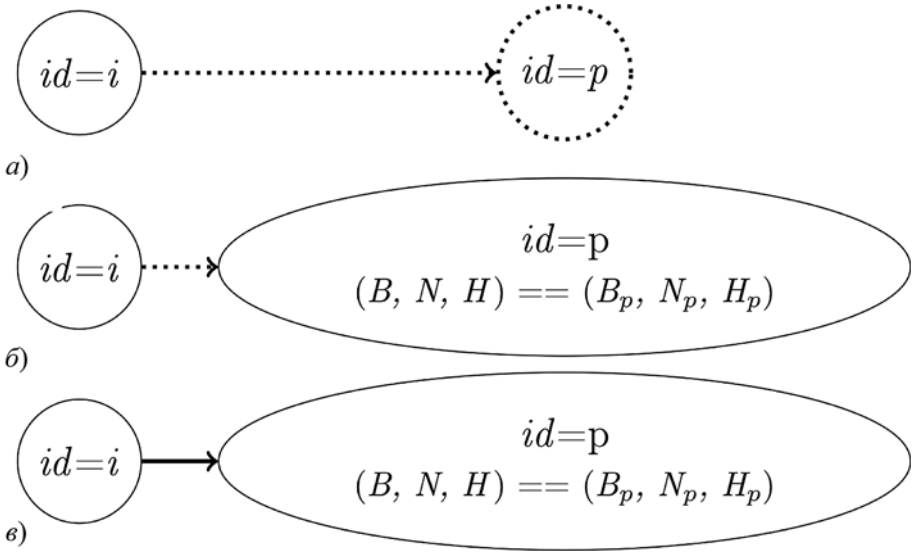

Рис. 7. Три ситуации, которые возникают после нажатия кнопки. Текущая вершина имеет метку $id = i$. Пунктирными линиями обозначены объекты, которые необходимо добавить в граф состояний

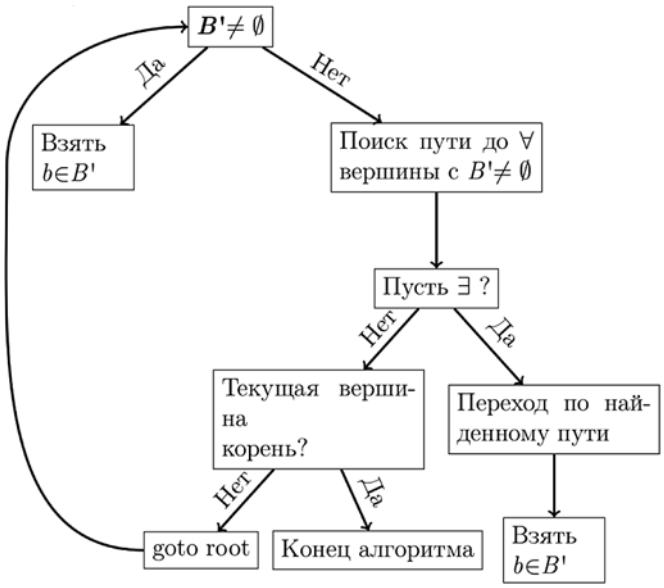

Рис. 9. Схема, описывающая правило выбора кнопки для нажатия

Рис. 10. Схема, описывающая правило выбора кнопки для нажатия

История нажатий

На рис. 11 (см. третью сторону обложки) изображена ситуация, при которой программа, реализующая алгоритм, дважды попала в вершину $id = i$. Находясь в первый раз в вершине $id = i$ и "нажав" кнопку b, программа оказалась в вершине $id = q$. Попав второй раз в вершину $id = i$ и "нажав" кнопку b, программа попала в другую вершину $id = p$. Если в данном случае добавить новое ребро, то получится, что из вершины $id = i$ будут выходить два ребра, которым приписана одна и так же кнопка. В этом случае возникает неоднозначность: если программа в третий раз попадет в вершину $id = i$, то куда будет сделан переход при нажатии кнопки b? Важной частью алгоритма является поиск пути до вершины, в которой есть ненажатая кнопка. В случае описанной неоднозначности становится затруднительно вычислять пути до вершины с ненажатой кнопкой.

Ситуация, изображенная на рис. 8 (см. третью сторону обложки), могла произойти в результате следующих трех причин.

1. Первая причина: на результат нажатия повлияло нажатие предыдущих кнопок. На рис. 8 (см. третью сторону обложки) переходу по ребру (i, q) предшествовала история нажатий кнопок [..., 4, 2], а переходу по ребру (i, p) — [..., 5, 2].

2. Вторая причина может заключаться в том, что веб-страница могла самопроизвольно измениться, например, мог встроиться новый URL. Причем в результате нажатия кнопки были проведены одни и те же действия, однако вследствие самопроизвольного изменения страницы, для алгоритма страница выглядит иначе с точки зрения параметров B, N, H. Под самопроизвольным изменением страницы имеется ввиду следующее. Изменения в веб-странице могут происходить в результате того, что программа "нажала" кнопку. Начиная с момента нажатия кнопки и до момента завершения ее нажатия считается, что страница меняется не самопроизвольно. Если страница изменяется вне указанного отрезка времени, то изменение страницы считается самопроизвольным.

3. Третья причина может заключаться в том, что со временем обработчик событий на кнопке начал работать по-другому. Такую функцию мог реализовать разработчик веб-сайта. Например, спустя 10 с после загрузки страницы функция начинает работать иначе. Пример самопроизвольного изменения изображен на рис. 12 (см. третью сторону обложки). С самопроизвольными изменениями состояния страницы можно бороться с помощью интерфейса MutationObserver. Этот интерфейс предписывается стандартом (https://dom.spec.whatwg.org/#mutationobserver, https://www.w3.org/TR/dom/#mutationobserver) и реализован в Firefox. Он позволяет наблюдать за любыми изменениями DOM-дерева. С помощью MutationObserver можно сделать так, чтобы при изменении DOM-дерева вызывались JavaScript-функции, которые написал разработчик алгоритма. Используя интерфейс MutationObserver, можно реализовать JavaScript-функцию, которая будет вызываться при изменения DOM-дерева. Данная функция должна пометить те узлы дерева, которые изменились самопроизвольно. В качестве меток, которые символизируют о том, что узел DOM-дерева самопроизвольно изменился, можно использовать атрибуты узлов в DOM-дереве. Если узел самопроизвольно изменился, то данному узлу добавляется заранее известный атрибут. Помеченные узлы должны игнорироваться при вычислении параметров B, N, H.

Опишем пример зацикливания алгоритма. Пусть будет иметь место следующее стечение обстоятельств. Пусть отмеченный на рис. 12 (см. третью сторону обложки) блок при каждом своем изменении будет содержать в себе ранее не встреченный на данной странице URL. Пусть страница содержит в себе кнопку, нажатие на которую не осуществляет каких-либо модификаций страницы. Как следствие, нажатие этой кнопки должно давать петлю в графе. Пусть блок изменяется самопроизвольно после нажатия кнопки, но до вычисления параметров B, N, H.

Последовательность действий, приводящая к зацикливанию, выглядит следующим образом:

1) нажатие кнопки b;
2) самопроизвольное изменение;
3) вычисление B, N, H;
4) нажатие кнопки b;
5) самопроизвольное изменение;
6) вычисление B, N, H;
7) ...

Граф состояний, соответствующий излагаемому примеру, отображен на рис. 13. Этот граф состояний представляет цепочку вершин, где вершина i связана выходящим ребром только с вершиной $i + 1$. Таким образом, получаем зацикливание.

Описанные выше проблемные вопросы удается разрешить, если использовать интерфейс MutationObserver. В случае использования интерфейса, после первого изменения блок будет игнорироваться. С учетом интерфейса MutationObserver можно считать, что вторая из представленных выше причин зацикливания не может привести к ситуации, изображенной на рис. 11. С первой и третьей причинами предлагается бороться следующим образом. Когда реализация алгоритма впервые оказывается в вершине $id = i$, после чего нажата кнопка b и реализация оказывается в вершине $id = q$, то ребру приписывается история длины ноль. Когда реализация алгоритма вновь попадает в вершину $id = i$, вновь нажимает кнопку b и оказывается в вершине $id = p$, то на ребрах удлиняется история. Пусть полная история нажатий кнопок имеет вид [..., 4, 2, b,..., 5, 2, b]. Если взять историю длины 1, то метки ребер будут одинаковыми и будут иметь значения (b, [2]). Если взять историю длины 2, то метки на ребрах становятся различными. Ребру (i, p) будет приписана метка (b, [5, 2]), а ребру (i, q) метка (b, [4, 2]).

Удлинение истории основывается на том, что результат нажатия кнопки определяется предыдущими нажатиями. Однако это предположение может быть неверно, поскольку могла измениться работа обработчика событий. Для этого предлагается ввести ограничение на максимальную длину удлинения истории. Пусть максимальная длина истории может равняться L, пусть была встречена неоднозначность, подобная описанной на рис. 11 (см. третью сторону обложки). Пусть процесс удлинения истории каждого ребра дошел до состояния, когда история каждого ребра имеет длину L и истории совпадают. В этом случае метка с кнопкой b должна быть удалена с того ребра, по которому не был сделан последний переход. В случае если $L = 1$, то в примере, представленном на рис. 11 (см. третью сторону обложки), должна быть удалена метка с ребра (i, q). Если на ребре не осталось других меток, то ребро должно быть удалено.

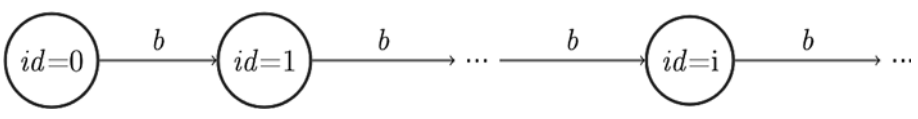

Рис. 13. Пример зацикливания

Использование историй усложняет операцию поиска пути до вершины. Рассмотрим операцию поиска пути. Пусть текущей вершиной в графе состояний является P_0. Пусть от начала работы программы и до момента попадания в вершину P_0 были нажаты кнопки b_{-N},..., b_0. Рассмотрим последовательность кнопок b_1,..., b_n. После нажатия каждой b_s текущей вершиной становится некоторая вершина P_s. Пусть из вершины P_n выходит ребро d, которому приписана история [q_1,..., q_m]. Если найдется такое j, для которого $b_{n-j}! = q_{m-j}$, то считается, что переход по ребру d невозможен и, как следствие, последовательность кнопок b_1,..., b_{n+1} не должна рассматриваться как путь для перехода в вершину с ненажатой кнопкой.

Следует также заметить, что при переходе к вершине с ненажатой кнопкой по некоторому пути очередной переход может привести в неожиданную (не предполагаемую) вершину. В этом случае, если необходимо, добавляется новое ребро и выполняется операция разрешения конфликтов с использованием удлинения истории. После этого необходимо заново запустить операцию поиска пути до вершины с ненажатой кнопкой (см. шаг 6. алгоритма).

Подробнее следует рассмотреть операции перехода в корень (перезагрузки страницы). При переходе в корень возможна одна из обозначенных далее трех ситуаций. Пусть страница была перезагружена и были вычислены значения B, N, H для страницы после перезагрузки. Пусть (B_{root}, N_{root}, H_{root}) представляют параметры корневой вершины.

1. $(B, N, H) = (B_{root}, N_{root}, H_{root})$. Новые параметры корня совпадают со старыми параметрами корня. В этом случае корень назначается текущей вершиной. Никаких изменений с перестройкой графа не осуществляется.

2. $(B, N, H) \neq (B_{root}, N_{root}, H_{root})$. При этом среди всех вершин графа не существует вершины, которой приписаны параметры (B, N, H). В этом случае старые параметры корня заменяются новыми параметрами.

3. $(B, N, H) \neq (B_{root}, N_{root}, H_{root})$, при этом нашлась вершина A, для которой $(B, N, H) = (B_A, N_A, H_A)$. В данном случае корневую вершину *root* необходимо удалить. После удаления корневой вершиной назначается вершина A. При этом необходимо осуществить перенос ребер со старого корня на новый. При переносе ребер могут возникнуть конфликтные ситуации.

Пусть существует вершина X и ребра (*root*, X) и (A, X). Пусть по обоим ребрам переход был сделан по кнопке b. Пусть на ребре (*root*, X) имеются истории h_1,..., h_n для кнопки *b*. Тогда все истории h_i должны быть перенесены на ребро (A, X). После переноса всех ребер с общей концевой вершиной переносятся ребра, не имеющие общих концевых вершин. Такие ребра переносятся по следующему правилу: исходная вершина *root* (*root*, Y) заменяется вершиной A. Входящие ребра также необходимо перенести. Для ребер с общей начальной вершиной (Z, *root*), (Z, A) необходимо перенести все истории с ребра (Z, *root*) на (Z, A). После переноса меток ребер, ребра (Z, *root*) удаляются. В случае с вершинами Y, такими что ребро (Y, *root*) существует, но ребра (Y, A) не существует, для всех ребер (Y, *root*) концевая вершина *Root* заменяется вершиной A.

Если при перезагрузке страницы пришлось обновлять параметры B, N, H в корне, то вероятно, у вершин, соседних с корнем, также придется обновить параметры. Если осуществлять обновление параметров только в корне, то может возникнуть ситуация, изображенная на рис. 14. Формально, вершины x^0 и x^1, $x \in [1—3]$ являются разными, поскольку у них различаются тройки параметров B, N, H. Однако до каждой из вершин x^0 и x^1, можно добраться из корневой вершины с помощью нажатия одной и той же последовательности кнопок.

Пример, представленный на рис. 14, иллюстрирует ситуацию, которая приводит к зацикливанию алгоритма. Предположим, что $b = b1 = b2$, а также в вершинах типа 2^n существует кнопка d (рис. 15).

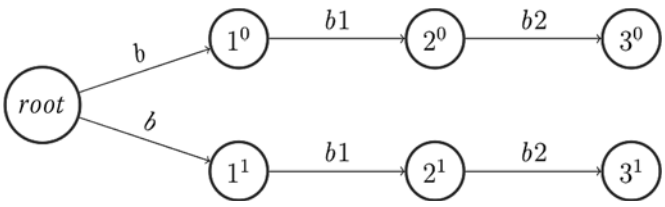

Рис. 14. Пример ситуации, при которой была выполнена операция обновления страницы, после чего последовало обновление параметров *B*, *N*, *H* в корне, но при этом не было обновления параметров в других вершинах графа

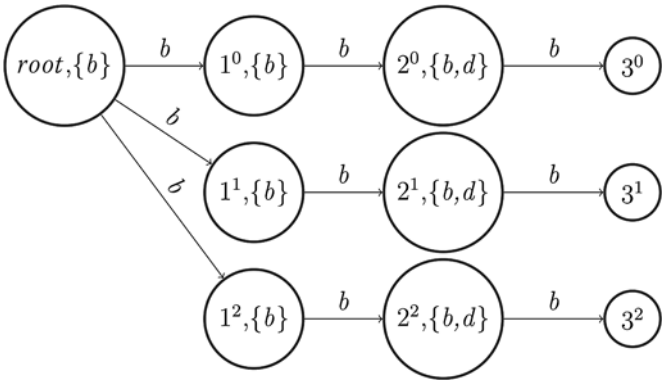

Рис. 15. Пример зацикливания алгоритма

Пусть программа, реализующая алгоритм, впервые загрузила страницу и "нажала" кнопку b три раза. В результате данной операции текущей вершиной окажется 3^0. В вершине 2^0 содержится кнопка d, которую программа должна "нажать". Для осуществления нажатия будет выполнена перезагрузка страницы, в результате чего текущей вершиной окажется корневая вершина. Предположим, что параметры корня и других вершин изменились. Программа, согласно алгоритму, проведет обновление параметров в вершине *root*, после чего попытается попасть в вершину 2^0. Однако при первом нажатии кнопки b программа попадает не в вершину 1^0, а в вершину 1^1. Произошел переход в непредполагаемую вершину. В этом случае запустится операция поиска вершины, в которой имеется ненажатая кнопка. В вершине 1^1 имеется кнопка b. Согласно алгоритму, будет проведено нажатие найденной кнопки. После первого нажатия в текущей вершине будет обнаружена еще одна кнопка b. Будет проведено еще одно нажатие кнопки b. В результате двух нажатий текущей вершиной окажется вершина 3^1, однако в вершине 3^1 отсутствуют кнопки. Как следствие, будет запущена операция поиска пути до ненажатой кнопки. В результате этой операции будет найден путь $root \to 1^1 \to 2^1$. Обновив страницу, программа попытается перейти по данному пути и вновь первый переход приведет в непредполагаемую вершину (не в 1^1, а в 1^2).

С учетом представленного примера (см. рис. 14, 15) предлагается следующее правило обновления параметров B, N, H в вершинах, соседних с корнем. Если при переходе текущей вершиной оказывается не ожидаемая вершина X, а вершина X', но при этом для перехода в вершину X программе потребовалось "нажать" последовательность кнопок B = [JUMP_ROOT, $b1$,..., bn] и, чтобы попасть в X', было выполнено нажатие такой же последовательности кнопок, то в этом случае предлагается правило: вместо добавления новой вершины и ребра выполняется обновление параметров B, N, H в вершине X, при этом текущей вершиной назначается вершина X.

Представленное выше соображение об обновлении параметров B, N, H в соседних с корнем вершинах исключает один из шаблонов зацикливания. Однако его использование не исключает всего набора ситуаций зацикливания. В связи с этим обстоятельством при реализации алгоритма на практике рекомендуется установить ограничение на максимальное число нажатий кнопок на странице.

Эвристика выбора кнопки

Пусть программа, реализующая алгоритм, находится в некоторой вершине графа и в этой вершине имеется множество ненажатых кнопок B'. Какую из кнопок следует выбирать так, чтобы обход страницы занял наименьшее время?

На рис. 2 (см. вторую сторону обложки) представлена кнопка "стрелка влево". Пусть программа N раз "нажала" кнопку "стрелка влево". Если алгоритм на-

жмет кнопку вне блока "Архив новостей", то данный блок исчезнет со страницы. В случае пропавшего блока, согласно алгоритму, программа в дальнейшем должна будет вернуться в то место, где кнопка "стрелка влево" была нажата N раз, и "нажать" ее еще один раз. В описанной ситуации появляется N лишних нажатий.

В связи с представленным примером предлагается, чтобы в программе сохранялся текст последней нажатой кнопки, и, при выборе очередной кнопки для нажатия среди кнопок множества B', будет выполняться поиск той, текст которой совпадает с сохраненным текстом. Если такая кнопка нашлась, то она выбирается для нажатия. Если такой кнопки не нашлось, то кнопки множества B' упорядочиваются по возрастанию редакторского расстояния до текста последней нажатой кнопки и выбирается кнопка с наименьшим расстоянием.

Разрастающиеся списки

На практике встречаются кнопки, которые можно нажать десятки тысяч раз. При этом, в результате очередного нажатия в веб-страницу будут встроены новые данные, а старые данные не будут удалены. Как следствие, веб-страница увеличивается с каждым нажатием кнопки, в результате чего веб-браузер начинает работать медленно. В этом случае не исключена ситуация, при которой закончится оперативная память. Пример такой кнопки приведен на рис. 1 (см. вторую сторону обложки). Нажатие этой кнопки встраивает в веб-страницу примерно 20 ссылок на публикации. Под ссылкой понимается гиперссылка на публикацию, картинка или какой-либо другой текст. При очередном нажатии старые ссылки не пропадают. Отмеченную кнопку можно нажать более 70 000 раз. Если не очищать веб-страницу, то после 70 000 нажатий в веб-странице окажется 1 400 000 ссылок. Такого количества будет достаточно, чтобы браузер начал работать медленно.

Для преодоления описанного затруднения необходимо в автоматическом режиме выделять части веб-страницы, которые можно удалить. В случае с примером относительно веб-страницы, изображенной на рис. 1, идеальным вариантом является усечение растущего списка ссылок. Следует отметить, что удаленный узел дерева может являться одной из кнопок, удалять которые нежелательно. Удаленный узел при этом может не являться кнопкой, однако может содержать в своих атрибутах какие-либо данные, необходимые для корректной работы той или иной кнопки. Удалять такие узлы тоже нежелательно.

Для разрешения вопросов, связанных с разрастающимися списками ссылок, применяется алгоритм поиска таких списков и их усечения. Представленный алгоритм никак не учитывает, является

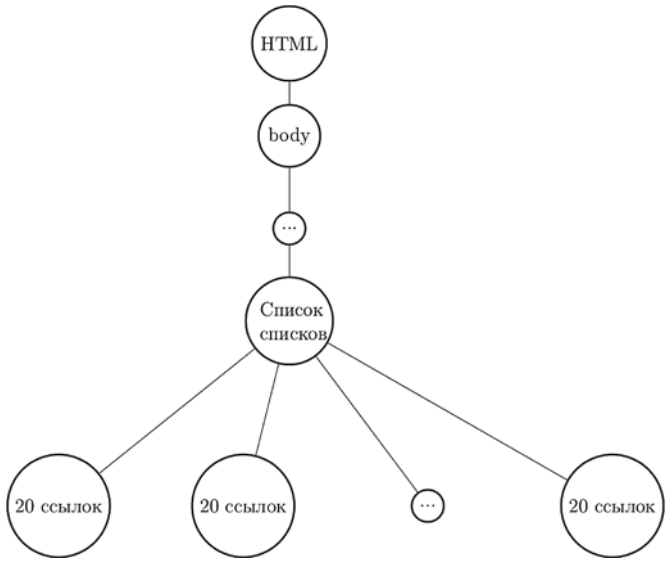

Рис. 16. DOM-дерево, соответствующее разрастающейся веб-странице

ли узел DOM-дерева кнопкой или нет. Однако на практике реализация алгоритма усечения списков, несмотря на свою простоту, не выбрасывает кнопки из дерева и усекает именно то, что необходимо.

Как правило, в результате нажатий кнопок веб-страница (DOM-дерево) разрастается не хаотичным образом, а в страницу (в DOM-дерево) встраиваются повторяющиеся поддеревья. В DOM-дереве, которое соответствует рис. 1 (см. вторую сторону обложки), это вершина U. При нажатии кнопки "загрузить еще" в потомки вершины U будут встраиваться списки из 20 ссылок на публикации. Схематично данная ситуация изображена на рис. 16.

HTML-разметка поддерева "список списков" приведена на рис. 17.

Первый элемент списка был добавлен при загрузке страницы, остальные элементы были добавлены в результате восьмикратного нажатия кнопки. Отличительной чертой списков является то обстоятельство, что у списка существует родитель, а непосредственные потомки родителя похожи друг на

```
<div class="b-list-normal">
  <div class="b-list-normal">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
  <div class="b-list" style="display: block;">20 ссылок</div>
</div>
```

Рис. 17. HTML-разметка поддерева "список списков"

друга. В рассмотренном примере родителем является элемент div с class="b-list-normal".

Пусть имеется функция F, которая получает на вход вершину дерева, а на выходе выдает число непосредственных потомков, которые похожи друг на друга. Применять эту функцию нужно следующим образом. После загрузки страницы данная функция применяется к каждому узлу дерева. Число, которое возвращает данная функция $F(X)$, приписывается узлу дерева X, обозначим это число в каждой вершине как N_0. После каждого нажатия кнопки должна вызываться функция F. Если для какого-то узла T выполняется $\frac{F(T)}{N_0} > \alpha$, то считается, что список увеличился достаточно сильно и его необходимо усекать. В этом случае все потомки T удаляются, однако сам T при этом не удаляется. В качестве константы α можно взять число два.

Определим функцию F. Пусть имеется функция $f(node)$, которая получает на вход вершину дерева и возвращает признаки вершины. Тогда по определению назовем вершину v похожей на своих соседей v_0, v_1, если $f(v) = f(v_0) = f(v_1)$. Если у вершины v не существует левого или правого соседа, то вершина v игнорируется. Пусть вершина T является родителем, $T_0,\ldots, T_n$ — непосредственные потомки T, а $F(T) = \sum_{i=1}^{n-1}\left[f(T_i) == f(T_{i-1}) \text{ and } f(T_i) == f(T_{i+1})\right]$.

Скобка $[f(x)]$ обозначает операцию, результатом которой является 1, если значение предиката внутри скобки *True*, и возвращает 0 в противном случае.

В качестве функции f можно взять функцию $f(node)$ = [имя_тега_node, имя_класса_node].

На этом завершается описание алгоритма, который предназначается для обхода веб-страницы. В результате обхода множества веб-страниц сформирована коллекция ссылок (URL) на публикации. Следующий раздел посвящен следующей задаче. Имеется URL, для которого известно, что в содержании веб-страницы содержится публикация, требуется отделить публикацию от всего остального, после чего она будет добавлена в БД.

Извлечение публикации

Сервис мониторинга публикаций предоставляет возможность выделения публикации следующими двумя способами: readability и "ручной".

Операция readabilty применяется к HTML-документам, в которых содержится статья. При выполнении данной операции из документа удаляется реклама, кнопки и меню на веб-сайте. Операция применяется в веб-браузерах и именуется "режим чтения". После удаления нежелательных элементов в документе должна остаться только статья.

Существует ряд реализаций readability. Известной реализацией является ARC90 [2]. На ее основе реализован режим чтения в Firefox [3] и в веб-браузере Safari. Реализация ARC90 и построенные на ее основе средства обрабатывают каждый HTML-документ независимо от других HTML-документов.

К недостаткам имеющихся на настоящее время реализаций можно отнести тот факт, что согласно алгоритму может быть допущена ошибка, в результате чего будет извлечена не статья. Например, если статья имеет небольшой размер, порядка двух предложений, то алгоритм может вместо статьи извлечь, например, информацию о копирайте, поскольку текст с копирайтом по объему больше, чем текст статьи. Другим недостатком является то обстоятельство, что угадав положение статьи верно, вместе со статьей, согласно алгоритму, может быть извлечен текст из меню веб-сайта и текст из кнопок.

К преимуществам readability можно отнести тот факт, что данный подход не требует настройки.

Другой способ выделения публикации — "ручной" — выделяет публикацию точно, не захватывая лишних частей веб-страницы. Однако данный метод требует ручной настройки. Обычно веб-страницы с публикациями обладают одним и тем же форматом. С точки зрения DOM-дерева фраза "обладают одним и тем же форматом" формулируется следующим образом. Пусть веб-страница содержит публикацию. Тогда существует вершина в DOM-дереве, в которой располагается статья. При этом путь от корня DOM-дерева до вершины с публикацией будет одинаков для всех веб-страниц. Как правило, на практике для одного веб-сайта встречается несколько типов веб-страниц, содержащих публикацию. Каждый тип обладает своим путем от корня до вершины с публикацией. На практике встречается от одного до четырех типов страниц.

Согласно данному подходу предлагается, чтобы программист потратил несколько минут на визуальный осмотр веб-сайта. После этого он должен выделить по одному представителю из разных типов веб-страниц и описать все пути от корня до вершины со статьей на языке XPATH. Пример пути от корня до вершины с публикацией на языке XPATH: //div[contains(@class,'b-inline-topics-box')]. Описанная процедура не является долгой и занимает порядка десяти минут.

С помощью алгоритма обхода веб-страницы и методов выделения публикации из веб-страницы было извлечено более трех миллионов публикаций с различных веб-сайтов. С помощью программного средства, описанного в работе [1], был осуществлен анализ публикаций на предмет эмоциональной окраски.

Программная реализация алгоритма и результаты тестовых испытаний

Функции сбора URL и выделения публикации из веб-страницы реализованы в рамках одной компьютерной программы contfetcher. Эта программа реализована на языке Python.

Программа contfetcher применяется для обработки одного веб-сайта. Для обработки нескольких веб-

сайтов необходимо запускать несколько экземпляров программы contfetcher. Для каждого веб-сайта в БД создаются свои экземпляры таблиц.

Программа contfetcher является многопроцессной. Существует главный процесс, который соединен коммуникационным каналом с СУБД и с процессами типа Worker. Главный процесс отвечает за запуск необходимого числа процессов типа Worker и за распределение заданий между процессами типа Worker. Под заданиями понимается, например, скачивание веб-страницы, которая соответствует заданному URL, и извлечение публикации из HTML-кода веб-страницы или обход веб-страницы, при котором будут извлекаться и сохраняться все встреченные URL. Если один из процессов типа Worker аварийно завершается, то главный процесс перезапустит процесс типа Worker. Визуальное представление компонентов программы дано на рис. 18.

В рамках программы contfetcher запускаются приложения. Каждое приложение можно включить и выключить. Для каждого включенного приложения при старте программы будут создаваться процессы типа Worker и при работе программы каждому процессу типа Worker будут высылаться задания для обработки. Если приложение выключено, то процессы типа Worker для данного приложения запускаться не будут.

В программе contfetcher существует три стандартных приложения.

1. Приложение traverse — обход веб-страницы с исполнением JavaScript. Данное приложение применяется для извлечения URL с веб-страницы, его имеет смысл применять на страницах, где для получения URL необходимо интерактивно взаимодействовать с веб-страницей. Примером такой страницы является https://ria.ru/lenta/. Для получения наиболее полной коллекции URL на этой странице необходимо нажимать кнопку "Загрузить еще". В рамках данного приложения веб-страница загружается с помощью веб-браузера Firefox. Следует отметить, что при использовании Firefox будет осуществляться ожидание загрузки всех CSS и JavaScript-файлов, в результате чего время загрузки страницы может занимать время, равное нескольким секундам. Для сравнения, при скачивании только HTML-кода веб-страницы, как правило, требуется время, равное десятым долям секунды.

2. Приложение readability — в процессе работы данного приложения из БД будут извлекаться URL, после чего для каждого URL будет осуществляться загрузка HTML-кода. Из HTML-кода в автоматическом режиме будет осуществляться извлечение публикации, название публикации, автор публикации, время публикации. Тело извлеченной статьи будет сохранено на жесткий диск. В БД будет сохранено название статьи, имя автора публикации, время публикации, путь до тела статьи на жестком диске и URL, из которого была извлечена публикация. Достоинством данного похода является то, что от пользователя не требуется никаких усилий для выделения статьи из HTML. К недостаткам можно отнести следующие:

- публикация может выделиться неточно, а именно — вместе с телом публикации могут быть захвачены элементы антуража веб-страницы (кнопки, меню и пр.);
- публикация может не выделиться, даже при условии, что на веб-странице она есть;
- публикация может быть выделена там, где ее нет, т. е. будет выделен "мусор".

Операция readability осуществляется с помощью веб-браузера Firefox. В Firefox данная операция называется режимом чтения.

3. Приложение grabber — данное приложение выполняет те же функции, что и readability, за исключением того, что правила для извлечения публикации описывает программист. Правила для извлечения

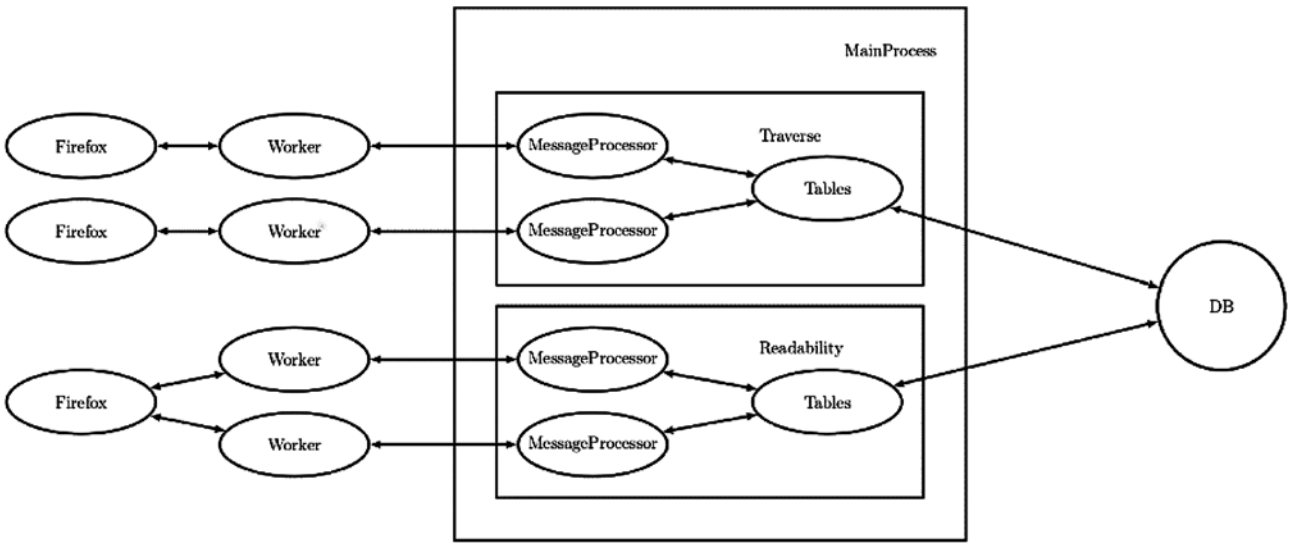

Рис. 18. Визуальное представление компонентов программы contfetcher

пишут на языке запросов к XML-документам — XPATH. На практике такие правила занимают несколько строк кода в конфигурационном файле. Использование данного похода предпочтительнее, чем readability, поскольку публикация будет извлекаться точно.

Кроме использования стандартных приложений для программы contfetcher существует возможность написания пользовательских приложений. При необходимости, функционал пользовательского приложения могут наследовать функции другого приложения.

Каждое приложение состоит из трех классов: MessageProcessor; Tables; Worker. Все три класса реализуются на языке Python. Перечисленные классы имеют следующее назначение.

1. Worker — данный класс запускается в рамках процесса типа Worker. Данный класс принимает задания на обработку от главного процесса. В процессе выполнения задания осуществляется загрузка вебстраницы и все необходимые операции, связанные с обработкой HTML или других данных, полученных от удаленного сервера. Результатом обработки задания является либо сообщение с полезными данными, либо сообщение с кодом ошибки.

2. Tables — в рамках данного класса должен быть написан код (SQL), с помощью которого будут созданы таблицы в БД. Через данный класс осуществляется также вставка данных в БД.

3. MessageProcessor — данный класс используется ядром программы contfetcher для отправления заданий в процессы типа Worker (в класс Worker) и для приема сообщений от процессов типа Worker. Данный класс должен вызывать методы класса Tables для вставки данных в БД.

На настоящее время сервис осуществляет мониторинг публикаций, в которых содержится название хотя бы одной из следующих организаций:

- МГУ имени М. В. Ломоносова;
- Институт динамики систем и теории управления имени В. М. Матросова СО РАН;
- Институт нефтехимического синтеза им. А. В. Топчиева Российской академии наук;
- Институт прикладной математики им. М. В. Келдыша;
- Институт цитологии РАН;

Упоминания научных организаций в СМИ

Организация	Число упоминаний			
	Отнесенных к положительному классу	Отнесенных к отрицательному классу	Отнесенных к нейтральному классу	Неоднозначных
МГУ имени М. В. Ломоносова	1959 (77,5)*	185 (7,3)	376 (14,8)	6 (0,2)
Институт динамики систем и теории управления имени В. М. Матросова СО РАН	1 (100,0)	0 (0)	0 (0)	0 (0)
Институт прикладной математики им. М. В. Келдыша	2 (100,0)	0 (0)	0 (0)	0 (0)
Институт цитологии РАН	3 (100,0)	0 (0)	0 (0)	0 (0)
Институт нефтехимического синтеза им. А. В. Топчиева Российской академии наук	2 (100,0)	0 (0)	0 (0)	0 (0)
Московский педагогический государственный университет	84 (88,4)	0 (0)	11 (11,5)	0 (0)
Российский государственный гуманитарный университет	180 (87,8)	0 (0)	25 (12,1)	0 (0)
ФГБНУ «Почвенный институт им. В. В. Докучаева»	0 (0)	0 (0)	0 (0)	0 (0)
ФГБНУ Институт машиноведения им. А. А. Благонравова РАН	1 (100,0)	0 (0)	0 (0)	0 (0)
ФГБНУ Научный центр неврологии	7 (100,0)	0 (0)	0 (0)	0 (0)
ФГБУН Физико-технический институт УрО РАН	0 (0)	0 (0)	0 (0)	0 (0)
Воронежский государственный университет	47 (87)	2 (3,7)	5 (9,2)	0 (0)
Полярный геофизический институт	0 (0)	0 (0)	0 (0)	0 (0)
Институт географии РАН	13 (81,25)	0 (0)	2 (12,5)	1 (6,25)
*В скобках указан процент упоминаний данного класса от общего числа упоминаний				

- Московский педагогический государственный университет;
- Российский государственный гуманитарный университет;
- ФГБНУ "Почвенный институт им. В. В. Докучаева";
- ФГБНУ Институт машиноведения им. А. А. Благонравова РАН;
- ФГБНУ Научный центр неврологии;
- ФГБУН Физико-технический институт УрО РАН;
- Воронежский государственный университет;
- Полярный геофизический институт;
- Институт географии РАН.

Сервис предоставляет возможность сформировать коллекцию публикаций, в которых встречается упоминание организации из приведенного списка, после чего провести автоматическую классификацию на три класса: "положительный", "отрицательный", "нейтральный". Если публикация относится к классу "положительный", то это означается, что она положительным образом влияет на формирование репутации организации. Отнесение к классу "отрицательный" означает, что публикация негативным образом сказывается на формирование репутации организации. Отнесение публикации к классу "нейтральный" означает, что она не влияет на формирование репутации.

В таблице приведены результаты мониторинга публикаций за 2016 г. Разъяснения требует столбец "неоднозначные". Это означает, что публикация влияет как положительно, так и отрицательно на формирование репутации. Метка "неоднозначно" не приписывается публикации при автоматической классификации публикаций. Данная метка существует, если эксперт вручную указал, что публикация улучшает и очерняет репутацию организации одновременно.

Заключение

В статье описан алгоритм, с помощью которого возможно реализовать обход веб-страницы путем нажатия кнопок, попадая при этом во всевозможные места веб-страницы. Изложенный алгоритм был реализован авторами и применяется в ИАС ИСТИНА для мониторинга веб-сайтов СМИ с целью извлечения публикаций.

Список литературы

1. **Васенин В. А., Роганов В. А., Дзабраев М. Д.** Методы автоматизированного анализа тональности текстов в средствах массовой информации // Программная инженерия. 2016. Т. 7, № 8. С. 360—372.
2. **ARC90,** readability. URL: http://code.google.com/p/arc90labs-readability.
3. **Mozilla** Firefox readability. URL: https://github.com/mozilla/readability

Methods and Means for Monitoring Publications in Mass Media

V. A. Vasenin, vasenin@msu.ru, Moscow State University, Moscow, 119234, Russian Federation,
M. D. Dzabraev, dzabraew@gmail.com, ISTINA Information System, Moscow, 119192, Russian Federation

Corresponding author:

Vasenin Valery A., Professor, Moscow State University, Moscow, 119192, Russian Federation,
E-mail: vasenin@msu.ru

Received on August 21, 2017
Accepted on September 07, 2017

Currently various applications encounter a task of extracting various data from the web sites. The first step to solve this task is extraction of all URLs from the web site being analyzed. Modern web sites are usually interactive, where interactivity means that the site can listen to events generated by user and respond to them. The most important event is clicking the left mouse button. To obtain the most complete collection of URLs one should click a certain sequence of buttons on the web page, which may cause a new block containing new URLs to be dynamically inserted. In other words, to obtain the most complete collection of URLs one should develop an algorithm that emulates actions of a user. This article presents model views, algorithms and software that emulates mouse clicks by a user. It also presents a business process model for algorithms and software using which one may automatically navigate within a page. Web pages may contain buttons of two types: clicking on the button either opens a new page, or modifies the current page by evaluating JavaScript. The navigating algorithm ignores the first type and only deals with the second type of buttons. The algorithm presented in this articles is intended to work with the following assumptions.

The implementation should automatically detect the buttons of second type on the page and automatically choose a sequence of buttons to be clicked on. The algorithm takes into account that as a consequence of clicking old buttons may disappear and new buttons may appear. If some button was present and then disappeared without being clicked, the memory of the implementation will contain a path using which the implementation will later come into state when this button was present and click it.

Keywords: *data extraction, web, readability, web-site traverse, web-page traverse, Javascript, Firefox*

For citation:

Vasenin V. A., Dzabraev M. D. Methods and Means for Monitoring Publications in Mass Media, *Programmnaya Ingeneria,* 2017, vol. 8, no.11, pp. 490—503.

DOI: 10.17587/prin.8.490-503

References

1. **Vasenin V. A., Roganov V. A., Dzabraev M. D.** Metodi avtomatizirovannogo analiza tonalnosti tekstov v sredstvah massovoi informacii (Methods of Automated Sentiment Analysis of Texts Published by Mass Media), *Programmnaya Ingeneria*, 2016, vol. 7, no. 8, pp. 360—372 (in Russian).

2. **ARC90** readability, available at: http://code.google.com/p/arc90labs-readability

3. **Mozilla** Firefox readability, available at: https://github.com/mozilla/readability

УДК 004.4’2 DOI: 10.17587/prin.8.504-510

П. А. Ченцов, канд. физ.-мат. наук, ст. науч. сотр., e-mail: chentsov.p@mail.ru,
Институт математики и механики им. Н. Н. Красовского УрО РАН, г. Екатеринбург

Система "Информационное пространство УрО РАН"

Рассмотрены особенности архитектурно-технологического подхода к построению систем связанных сайтов на примере информационной системы, призванной обслуживать нужды Уральского отделения Российской академии наук. Данный подход может быть распространен и на другие информационные интернет-проекты. К основным его особенностям относятся: высокая степень повторного использования кода и автоматическая генерация HTML-страниц.

Ключевые слова: *информационная система, результаты научно-технической деятельности*

Введение

С каждым годом информационные системы занимают все большее место в общественной жизни [1, 2]. К числу таких систем относят, например, поисковые системы, интернет-банки, социальные сети, информационные системы, сопровождающие торговую деятельность, различные бытовые услуги и др.

Основное направление работы Российской академии наук (далее — Академия или РАН) — создание новых знаний, информации и их активное использование. Как следствие, РАН имеет большую потребность в информационных ресурсах. Спектр научных исследований, которые проводятся в Академии, результатов их выполнения, а также различных мероприятий, достаточно широк. Сюда можно отнести публикации, такие как научные статьи, книги, пособия, тезисы в материалах конференций, научно-технические отчеты и некоторые другие результаты. В рамках исследований создается различное программное обеспечение. Сотрудники РАН участвуют с докладами в конференциях, осуществляют научное руководство студентами и аспирантами. Создаются выставочные образцы, с которыми сотрудники участвуют в выставках. Сотрудники регистрируют патенты, снаряжают экспедиции. Иными словами, производится большой объем информации, требующей тщательного учета и анализа.

Часть информации по публикациям содержится в существующих системах цитирования публикаций, таких как Web of Science, Scopus, Академия Google, elibrary и некоторых других. Одна из сложностей их использования состоит в том, что такие источники не являются традиционной общедоступной реляционной базой, и как следствие, на их основе трудно строить какие-либо автоматические отчеты в заданном формате с заданной полнотой требуемой информации. Кроме того, зачастую востребованная информация появляется несвоевременно, со значительной задержкой, а сам ее состав неполон.

Кроме информации о результатах научной деятельности существует большое число задач, призванных обслуживать административно-управленческие потребности Академии. Сюда можно отнести, например, системы работы с документами, кадровую систему, систему поддержки обучения в аспирантуре, систему поддержки регистрации программного обеспечения и др.

Принимая во внимание отмеченные выше сложности, в Уральском отделении РАН (УрО РАН) была создана информационная система, решающая многие из указанных задач. Фактически, цель данной статьи состоит в ознакомлении читателя с одним комплексным подходом к разработке сложных информационных систем с возможностью быстрого расширения их функций в условиях относительно малого числа исполнителей, а также в его ознакомлении с интересными особенностями одного из вариантов реализации такой системы.

Постановка задачи

Современные информационные системы в области библиометрии, наукометрии и другие системы подобного назначения строят в виде интернет-приложений, что, как правило, обеспечивает возможность доступа к ним с любого компьютера и мобильного устройства без установки дополнительного программного обеспечения. Такой подход должен быть применен при разработке информационного пространства УрО РАН, представленного в данной публикации.

В плане пользовательского интерфейса к информационным ресурсам такого пространства можно сформулировать следующие требования. Система должна правильно функционировать на устаревших компьютерах с низким разрешением экрана и интернет-браузерами старых версий. Кроме того, страницы должны правильно отображаться на мобильных устройствах. Следует обеспечить полноту

информации, возможность получать развернутую информацию по выбранным объектам.

Все сайты системы должны быть построены вокруг одной базы данных. Очень важно иметь консистентные данные, связанные реляционными отношениями, так как на основе этих данных будут формироваться различные отчеты, проводиться оценка результативности как персональной, так и коллективной деятельности, а также многое другое. Доступ к информации должен быть дифференцирован и осуществляться в соответствии с уровнем доступа пользователя. Регистрацию пользователей должны проводить администраторы. Учитывая большое число потенциальных пользователей и территориально-распределенный характер организаций Академии, можно сделать вывод, что один человек не сможет своевременно добавлять в систему пользователей с должным уровнем верификации личности. Таким образом, система администрирования должна быть двухуровневая — главный администратор, выполняющий общее администрирование, и назначаемые им администраторы институтов, поддерживающие в актуальном состоянии информацию об инфраструктуре института и его сотрудниках. Целесообразным представляется делегирование части администраторских полномочий сотрудникам отдела кадров.

Все администрирование должно проводиться через интернет-браузер. Такой подход позволяет разрешать различные трудности даже в том случае, если администратор находится вне рабочего места. Кроме того, появляется возможность, при надобности, легко делегировать полномочия администратора какому-либо сотруднику (например, если администратор уходит в отпуск).

Аутентификация должна проводиться с использованием ActiveDirectory, что обеспечивает возможность использования пары логин — пароль не только в пределах информационного пространства, но и для других программ и систем. Помимо этого, следует предусмотреть базовую аутентификацию — на случай, если сотрудник по каким-либо причинам не желает использовать доменный логин, либо в случае, когда сотрудник не является работником Академии, но информационная запись о нем оказывается востребованной некоторыми функциями системы.

Реализация системы

Одним из основополагающих факторов при выборе технологии разработки явилось незначительное число разработчиков. По большей части система разрабатывалась одним человеком, что накладывало серьезные ограничения на выбор технологии.

Существует ряд подходов к реализации такого рода информационных систем. Например, это ASP.NET MVC [3], Zend Framework. Основная идея подхода в этих системах состоит в разделении представления (непосредственно HTML-страницы), модели (данные приложения) и контроллера — элемента программы, который выполняет все действия. Содержимое большинства страниц информационного пространства часто зависит от прав доступа пользователя. Следовательно, может существовать большое число вариантов представления одной и той же страницы и соответствующей ей модели. В таком случае возникает большое количество многократно повторяемой работы. В ASP.NET Forms [4] есть проблемы с повторным использованием кода, что представлялось очень важным. Инструментальное средство Drupal [5] громоздкое, требует длительного освоения и сильно нагружает сервер активным обменом с базой данных. Для решения поставленной задачи была разработана собственная платформа, названная MVE (Model/View/Events), представленная в работах [6, 7]. Эта платформа базируется на свободно распространяемом программном обеспечении Apache, PHP, MySQL. Данная тройка имеет устойчивые позиции в современном мире серверного программного обеспечения. Эти средства позволяют строить производительные и масштабные информационные системы. Система MVE позволила свести к минимуму число разработчиков при малом времени достижения результата. Это стало возможным благодаря эффективному механизму автоматической генерации страниц и высокой степени повторного использования кода.

В ходе анализа задачи был установлен факт, что большинство страниц зависят от двух параметров — от прав доступа пользователя и способа представления информации. Что касается прав доступа, то они представлены ролями двух типов: роли, связанные с текущей должностью; глобальные роли. Дело в том, что в Академии часто встречается совместительство. Один и тот же человек может работать в двух местах, причем иногда в разных институтах. Соответственно, объем и состав предоставляемых информационных услуг различается. Особенно это заметно в том случае, если на одном из мест сотрудник занимает руководящую должность. Таким образом, часть ролей связана с должностью. Некоторые роли не требуют жесткой привязки к должности — это, как правило, роли, связанные с сопровождением системы в целом.

Информация и функции, предоставляемые на странице, могут различаться в зависимости от контекста, в котором страница используется. Для понимания такого подхода проще всего привести пример. Рассмотрим страницу с таблицей публикаций. Дело в том, что существуют разные виды таблиц публикаций, к числу которых относятся публикации сотрудника, подразделения, института, отделения РАН. Могут встречаться и такие варианты — список публикаций, входящих в библиографию статьи или книги, и при этом зарегистрированных в информационном пространстве. Такой список удобно использовать при знакомстве с научными работами.

Подобные страницы различаются не только содержимым. Они могут включать различные функции, управляющие элементы. Во многих классиче-

ских подходах к разработке интернет-приложений возникает необходимость создавать те или иные конструкции (представления, контроллеры) для каждого случая отдельно. Одна из причин такого положения обусловлена тем, что, как правило, сами страницы (представления) разрабатывают вручную. В системе MVE предложен несколько иной подход. Программист работает с объектной моделью и может не обладать знаниями о HTML и браузерах. Реакции пользователя обрабатываются в специальных методах обработки событий. Данные хранятся в модели, автоматически восстанавливаемой из post-параметров, что избавляет разработчика от необходимости восстанавливать post-параметры вручную. Сами страницы формируются автоматически. Такой подход позволяет легко получать разные варианты страниц. Например, чтобы сделать недоступными для редактирования данные публикации, достаточно присвоить значение false свойству ReadOnly объекта публикации. По умолчанию все вложенные объекты также автоматически примут значение false данного свойства. Все конструкции (кнопки, поля ввода, флажки), связанные с редактированием, либо исчезнут, либо станут неактивными. Таким образом, изменив всего одну строку, можно получить значительно отличающуюся от исходной страницу. Для работы с объектами доступны такие известные программистам свойства, как Visible и ReadOnly. Кроме того, реализован ряд настроек, призванных существенно облегчить работу программиста.

Отмеченные выше особенности MVE-подхода далеко не полностью описывают возможности повторного использования программного кода. Иногда для ускорения разработки и упрощения архитектуры необходимо поместить содержимое одной такой страницы в содержимое другой. Например, список сотрудников может иметь множество контекстов использования, один из которых — список авторов публикации. Возникает потребность расположить этот список на странице с данными публикации. Такая задача также решена в системе MVE. Объекты, которые упоминались выше, соответствуют не целым страницам, а лишь участкам содержимого. Такие конструкции называются юнитами и могут вкладываться одна в другую, давая на выходе требуемую страницу. На рис. 1 приведен упрощенный пример страницы публикации. Слева располагается объектная модель, справа — автоматически сформированная на ее основе HTML-страница. Безусловно, данный пример не показывает всех возможностей платформы (например, работы с объектами размещения, валидаторами, управляющими конструкциями, отличными от поля ввода и кнопки, и др.), однако он позволяет получить общее представление о них.

Безусловно, подобный подход позволяет строить только шаблонные интерфейсы, хотя и со значительными возможностями кастомизации. Интерфейсы произвольного вида строить на этой основе не удастся, однако в подобных системах потребность в этом не возникает. Основных путей кастомизации два — параметры объектов и CSS-стили. Первыми пользуются разработчики. Вторые определяют внешний вид, стиль отображения страниц. Их формирует дизайнер. Параметры объектов позволяют в некоторой степени менять их вид. Обычно это выбор варианта дизайна объекта из нескольких предустановленных цветовых вариантов. Параметры представляют механизмы, позволяющие группировать объекты, выравнивать

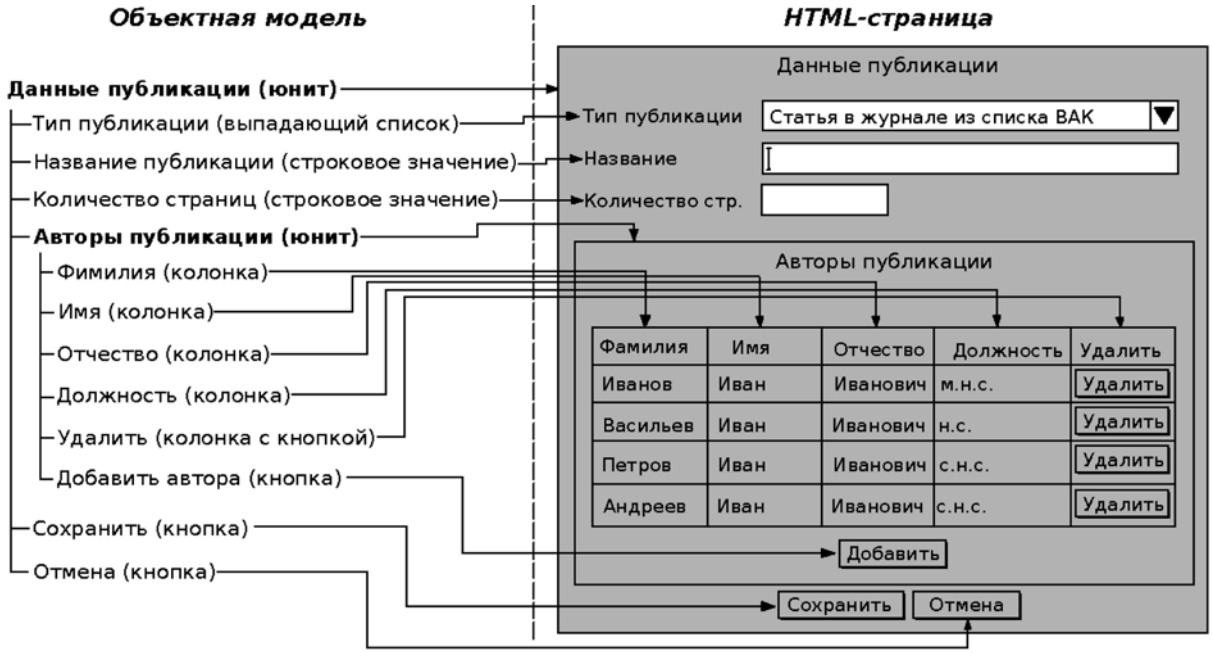

Рис. 1. Объектная модель и автоматическая генерация страниц

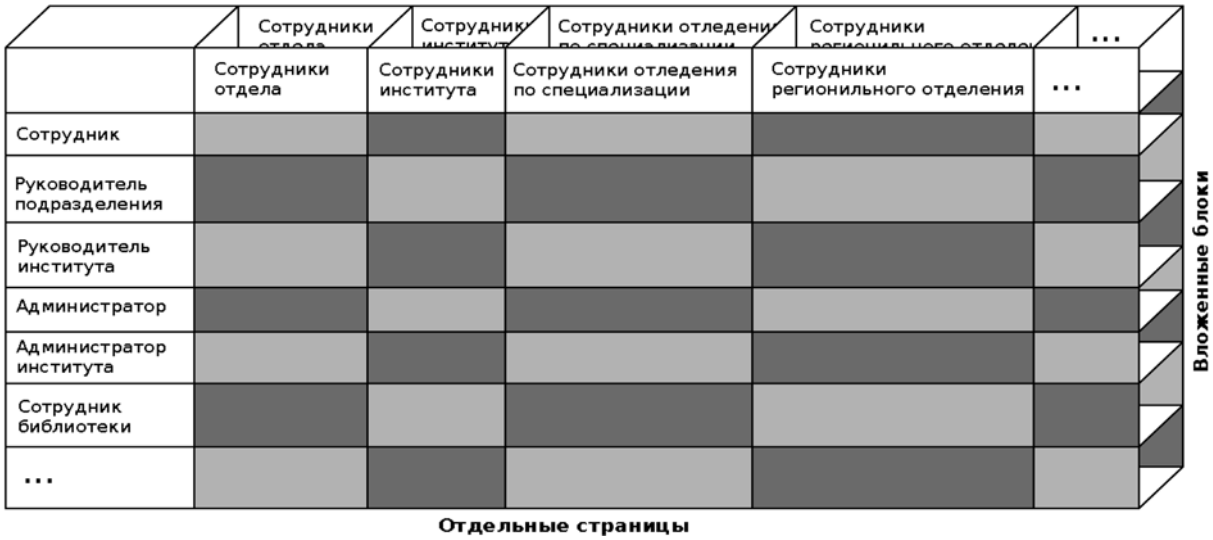

Рис. 2. Варианты использования объекта *Список сотрудников*

один относительно другого, менять возможности доступа для редактирования. Использование CSS-стилей позволяет более тонко настроить внешний вид объектов и страниц (цвета, шрифты, отступы и др.). Главное в таком подходе — разработчик избавляется от значительной рутинной деятельности, связанной с построением страниц. В ходе такой работы подчас возникают незначительные ошибки. Иногда эти ошибки только искажают интерфейс, однако в некоторых случаях они могут препятствовать штатному режиму функционирования системы.

Для обобщения изложенной информации рассмотрим рис. 2, на котором отображены различные варианты использования объекта *Список сотрудников*.

На рис. 2 видно, как велико может быть разнообразие информационных конструкций (включая HTML-код страницы), порождаемых одним и тем же объектом. При этом отметим, что некоторые клетки таблицы (информационные конструкции) могут совпадать, т. е. иметь одну и ту же реализацию.

Пользователь и ActiveDirectory

Система авторизации ActiveDirectory производства Microsoft в настоящее время получила широкое распространение. Многие современные программы и сервисы используют ее для аутентификации и авторизации, что позволяет пользователю иметь одну пару логин — пароль для доступа к разным ресурсам. Академия не стала исключением — здесь также активно применяется ActiveDirectory. Однако ActiveDirectory требует администрирования. Иными словами, для новых сотрудников, для которых в информационном пространстве следует создавать записи, их нужно создать и в ActiveDirectory. Кроме того, в дальнейшем следует синхронно выполнять все изменения. В информационном пространстве для этих целей разработан специальный механизм, позволяющий выполнять всю работу с ActiveDirectory непосредственно из сервера сайта администрирования. Все действия, включая создание записи и назначение начального пароля, выполняются через web-интерфейс, а именно, через страницы сайта администрирования. Необходимость работы непосредственно с ActiveDirectory возникает достаточно редко, например, при создании нового сервера для института. На рис. 3 изображена схема взаимодействия указанных систем.

Данный подход позволяет сократить число администраторов ActiveDirectory или вообще исключить их, совместив самые необходимые их обязанности с обязанностями администратора информационного пространства. Кроме варианта с ActiveDirectory предусмотрена также базовая аутентификация пользователей, при которой логин и пароль (в зашифрованном виде) сохраняются в базе данных системы.

Меню и виртуальные сайты

Для осуществления навигации по ресурсам информационного пространства с учетом авторизации пользователей была разработана библиотека для построения предназначенного для этих целей меню, выходящая за рамки механизмов, предоставляемых MVE. Особое внимание при этом было уделено вопросам информационной безопасности. Учитывая тот факт, что система разрабатывалась для построения семейства сайтов, требовалось обеспечить возможность работы этих сайтов с одними и теми же библиотеками и базой данных. С учетом изложенных соображений было принято решение использовать технологию виртуальных сайтов, позволяющую существенно упростить разработку меню. Данная технология используется в ряде существующих систем построения интернет-приложений, и она хорошо себя зарекомендовала. Согласно этой технологии,

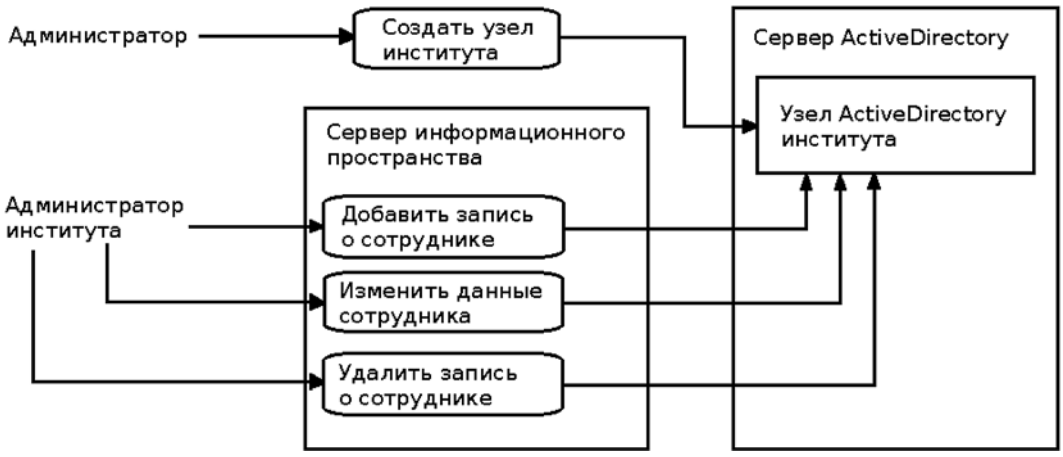

Рис. 3. Актуализация состояния сервера ActiveDirectory

внутри сайта выделяются папки, каждая из которых соответствует своему виртуальному сайту. При построении меню для текущей страницы проводится анализ ее адреса, из которого определяется, какому сайту она соответствует. Далее построение меню осуществляется именно для этого сайта.

В отличие от таких систем, как Drupal, сайты информационного пространства не хранят информацию о меню и архитектуре страниц в базе данных, что позволяет снизить нагрузку на процессор и базу данных. Конфигурируется меню в выделенных файлах проекта таким образом, что для каждого сайта такой файл свой. Несмотря на различия в меню сайтов, эти меню должны содержать еще и некоторые общие функциональные возможности. Например, элементы меню аутентификации или страницы с информацией о текущем сотруднике. В системе построения меню предусмотрена возможность добавления глобальных, общих для всех сайтов, элементов меню.

Главное преимущество используемой для разработки меню технологии состоит в цельности решения. Все сайты в любом случае используют один и тот же программный код. Отсутствует возможность ситуации, когда библиотеки одного сайта обновлены, а другого — нет. Не возникают вопросы с разграничением зон ответственности. Все файлы, за которые несет ответственность разработчик некоторого сайта, локализованы либо в папке этого виртуального сайта, либо в хранилище файлов объектов. Однако и в таком хранилище у каждого файла есть ответственный за него программист.

Кроме перечисленных выше положительных особенностей, такая авторская разработка на настоящее время представляется достаточно защищенной от взлома. Известны факты, что во многих случаях взлом осуществляется с учетом уязвимостей платформы, на которой разработан сайт или интернет-приложение. Таким образом, хакеру достаточно выяснить информацию о такой уязвимости и грамотно воспользоваться ею. Безусловно, разные системы разработки имеют разный уровень надежности, но хакеры в любом случае имеют большие возможности их изучения. В данном случае платформа разрабатывалась под проект, иных реализаций нет, как следствие, с точки зрения взлома она представляется черным ящиком. Фактически потенциальные возможности взлома могут присутствовать, но найти их — не простая задача.

Информация, регистрируемая в системе

Одна из основных функций информационного пространства — сбор информации о результатах деятельности сотрудников УрО РАН. С ней неразрывно связана другая функция — предоставление этой информации сотрудникам Академии и ФАНО. Во-первых, эта информация позволяет судить о результативности работы сотрудников и институтов. Во-вторых, все сотрудники получают доступ к информации о проводимых разработках. Что касается научных публикаций, то здесь, кроме информации библиографического характера, хранятся также и полные тексты работ, доступные для загрузки на компьютеры сотрудников. Работает полнотекстовый поиск. Все это существенно упрощает взаимодействие между структурными подразделениями и институтами УрО РАН, обеспечивает большие возможности для ознакомлении с работами коллег.

В системе осуществляется регистрация следующих видов результатов персональной научно-технической деятельности:

- научные публикации;
- участие в конференциях;
- участие в выставках;
- объекты интеллектуальной собственности;
- научно-технические отчеты;
- научное руководство аспирантами;
- научное руководство диссертационными работами, выпускными работами магистров и бакалавров.

Наиболее востребованным и проработанным является комплекс страниц для автоматизированной работы с публикациями. Следует заметить, что для

всех вводимых в систему публикаций выполняется их верификация сотрудниками библиотеки института. Каждая новая публикация попадает в список, доступный сотруднику библиотеки института, и покидает его уже после отметки о факте ее верификации. Данные самой публикации после этого становятся недоступными для редактирования.

Система позволяет формировать различные типы отчетов по отдельным сотрудникам, подразделениям, институтам. В отчетный период такая функция значительно упрощает работу ученого секретаря.

Особенности системы

В ходе работы была построена не просто информационная система, а своего рода конструктор для быстрого создания сайтов УрО РАН. Так как основные конструкции уже реализованы, разработка новых сайтов сводится по большей части к двум составляющим: добавление в существующие объекты новых режимов и реализация объектов, обладающих спецификой для сайта. Что касается добавления новых режимов, то, как правило, делается это достаточно просто и быстро. Например, списки сотрудников отдела, отделения или института различаются только заголовком и SQL-запросом к базе данных.

Некоторые принципы, по которым разрабатывалась система, несколько отличаются от наиболее распространенных в индустрии интернет-приложений. Одно из основных отличий — на страницах информационного пространства в большинстве случаев предоставляется максимально развернутая информация, в то время как во многих современных системах разработчики наоборот пытаются минимизировать объем страницы.

Например, выбор тех или иных информационных объектов принято проводить через выпадающий список. Вместе с тем длина строки в таком списке зачастую недостаточна для четкой идентификации выбираемого объекта. При этом если число строк начинает превышать 40...50, то поиск нужного значения может быть затруднен. Как правило, в таком случае используется строчка фильтра, но она не всегда позволяет изменить ситуацию к лучшему. Например, если точное название объекта неизвестно или известно не полностью. Рассмотрим выбор сотрудника, например, при формировании списка авторов статьи. Во-первых, могут быть совпадения фамилий и имен. Во-вторых, в Академии часто возникают ситуации, когда сотрудники по объективным причинам не регистрируют свои работы сами, а делегируют эти полномочия кому-либо другому, кто может не владеть точной информацией по авторам. В этом случае к авторам могут возникнуть вопросы. Их контакты приходится искать отдельно. Для разрешения этих вопросов в информационном пространстве список сотрудников для выбора (как и некоторые другие конструкции, например публикации) представляет собой отдельную страницу с информативной многоколоночной таблицей, постраничным перелистыванием, с фильтром, с возможностью выбрать одного (кнопкой в строке) или нескольких (флажками в строках) сотрудников. Предусмотрена также возможность перехода к странице с информацией по соответствующему сотруднику посредством нажатия определенной кнопки в любой строке таблицы.

Заключение

В ходе работы над информационным пространством УрО РАН была построена платформа разработки MVE [6, 7], обеспечившая возможность выполнения заданного объема работ. Для ее реализации был сформирован общий архитектурно-технологический подход, позволивший в большинстве случаев исключить повторные участки программного кода и обеспечивший значительный задел для разработки новых сайтов УрО РАН. Основная новизна проделанной на этом направлении работы состоит именно в применяемой технологии разработки. На ее основе построены и запущены в эксплуатацию основные сайты системы, включая сайт администрирования, отображения результатов научно-технической деятельности, аспирантуры.

В плане интерфейса применены некоторые приемы, повышающие информативность системы и облегчающие работу с ней. На первый взгляд, может показаться, что страницы перенасыщены информацией, но следует учитывать, что данная система — не социальная сеть или интернет-магазин. Полнота и точность информации здесь имеют приоритетное значение.

Следует также упомянуть о практическом значении разработанной информационной системы. Наиболее важная ее функция — сбор и накопление информации о результатах научной деятельности сотрудников, а также формирование на основе указанной информации различных отчетов. Учитывая специфику научной работы, такая система сильно облегчает оценку работы как отдельных сотрудников, так и подразделений и институтов УрО РАН, что является актуальной задачей.

Работа выполнена при поддержке программы Президиума УрО РАН 15-7-1-19.

Список литературы

1. **Титоренко Г. А.** Информационные системы в экономике. М.: ЮНИТИ-ДАНА, 2008. 463 с.

2. **Карр Н. Дж.** Великий переход: что готовит революция облачных технологий. М.: Манн, Иванов и Фербер, 2013. 272 с.

3. **Фримен А., Сандерсон С.** ASP.NET MVC 3 Framework с примерами на C# для профессионалов, 3-е изд. М.: Вильямс, 2012. 672 с.

4. **Мак-Дональд М., Фримен А., Шпушта М.** ASP.NET 4 с примерами для профессионалов. 4-е изд. М.: Вильямс, 2011. 1424 с.

5. **Мелансон Б., Нордин Д., Луиси Ж.** и др. Профессиональная разработка сайтов на Drupal 7. СПб.: Питер. 2013. 688 с.

6. **Ченцов П. А.** Платформа разработки интернет-приложений MVE // Программная инженерия. 2014. № 9. С. 13—22.

7. **Ченцов П. А**. Платформа разработки Web-приложений MVE // Труды XX Всеросс. науч.-метод. конф. Телематика 2013. СПб, 2013. Т. 2. С. 292—293.

Information Space of the Ural Branch of RAS

P. A. Chentsov, chentsov.p@mail.ru, Yekaterinburg, 620990, N. N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences (IMM UB RAS)

Corresponding author:

Chentsov Pavel A., Senior Researcher, Yekaterinburg, 620990, N. N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences (IMM UB RAS),
E-mail: chentsov.p@mail.ru,

Received on August 17, 2017
Accepted on September 01, 2017

Information technologies are used in many fields of human activity. They greatly simplifies many of the processes. Russian Academy of Sciences has a great need for various information systems. The Academy creates scientific knowledge, which requires accounting and systematization. There is also need to build various reports on the basis of recorded data. There are many different auxiliary tasks, which can also be automated using information systems. Such systems are built on web-based internet technology. It allows to use them without installing additional software on computers and mobile devices. The article discusses one approach to the development of a system of sites, combined into a single information space. The implementation of the system "Information space of the Ural branch of RAS" is considered. Key features are high degree of code reuse and automatic generation of HTML pages.

Keywords: *web application, information system, scientific and technical activities*

Acknowledgements: *This work was supported by the programm of the Presidium Ural Branch of RAS 15-7-1-19.*
For citation:

Chentsov P. A. Information space of the Ural branch of RAS, *Programmnaya Ingeneria*, 2017, vol. 8, no. 11, pp. 504—510.

DOI: 10.17587/prin.8.504-510

Refrences

1. **Titorenko G. A.** *Informacionnye sistemy v jekonomike* (Information systems in economy), Moscow, JuNITI-DANA, 2008, 463 p. (in Russian).
2. **Carr N. G.** *The Big Switch: Rewiring the World, from Edison to Google*, W. W. Norton & Company, 2008. 278 p.
3. **Frimen A., Sanderson S.** *Pro ASP.NET MVC 3 Framework*, Apress, 2011, 852 p.
4. **Mak-Donal'd M., Frimen A., Shpushta M.** *Pro ASP.NET 4 in C# 2010*, Apress 2010, 1616 p.
5. **Melanson B., Nordin D., Luisi Zh**. et. al. *The Definitive Guide to Drupal 7*, Apress 2011, 1112 p.
6. **Chentsov P. A.** Platforma razrabotki internet-prilozhenij MVE (Internet applications development system MVE), *Programmnaya ingeneria*, 2014, no. 9, pp. 13—22 (in Russian).
7. **Chentsov P. A.** Platforma razrabotki Web-prilozhenij MVE (Web-applications development system MVE), *Proceedings of XX conf. Telematica 2013*, Saint Petersburg 2013, vol. 2, pp. 292—293. (in Russian).

УДК 004.942 DOI: 10.17587/prin.8.511-523

А. А. Артемов, инженер, соискатель, e-mail: artemsince2@yandex.ru, АНО Центр проблем стратегических ядерных сил Академии военных наук, г. Юбилейный, Московская обл.,
А. В. Галатенко, канд. физ.-мат. наук, ст. науч. сотр., e-mail: agalat@msu.ru,
Механико-математический факультет МГУ имени М. В. Ломоносова

Моделирование процесса эволюции содержания информационного пространства социума (Мем-грамм-модель)

Представлены модель и алгоритм для прогноза числа копий мема в будущем содержании информационного пространства социума (ИП). Глубина прогноза модели составляет одну календарную неделю на каждые три значения числа копий мема в предыдущие три недели. Модель является статистической и создана на основе анализа изменений содержания русскоязычного сегментам ИП с мая по август 2014 г., представленного коллекцией изменяющихся во времени корпусов текстов, сформированных из 300 000 наиболее цитируемых в социальных сетях русскоязычных публикаций различных интернет-СМИ. Приведены оценки качества модели, определяющие вероятность ошибки прогноза числа копий мема. Особенностью предложенного алгоритма является возможность его применения для прогноза числа копий мема для любого ИП, близкого по размеру и разнообразию русскоязычному сегменту ИП.

Ключевые слова: *мем-грамм-модель, языковая модель, биграмма, мем, информационное пространство, распределение Su-Джонсона, теорема о плотности распределения числа копий мемов, статистическая модель, алгоритм прогноза числа копий мемов*

Введение

На практике хорошо себя зарекомендовали два подхода к построению языковых моделей представления знаний (далее — языковая модель), основанных на анализе текстов на естественном языке с использованием методов машинного обучения. Первым из них является подход, основанный на анализе н-грамм — лемматизированной (в исходной форме) совокупности слов, полученных из текстов [1]. Вторым является подход на основе векторов, представляющих связь выбранного слова с контекстом (последовательностью нелемматизированных слов, рассматриваемых как уникальная языковая единица). Наиболее успешным примером векторной модели является модель word2vec [2]. В моделях знаний предлагается функция, способная предсказывать вероятность того, что определенная последовательность слов является допустимой (вероятной) для данного языка. Соответственно, в общем случае языковая модель представления знаний задается двойкой элементов $\langle C_T, \Pr(w_i)\rangle$, где C_T — корпус текстов T, представленных мультимножеством словосочетании w; Pr — функция вероятности, которая ставит в соответствие набору элементов $w_i = w_{n-1}w_n$ из мультимножества T число так, что $Pr : N[w_i] \to [0;1] | \forall w_i \in C_T$, $N[w_i] \in \mathbb{N}$. Базовая н-грамм-модель для биграмм представляется в следующем виде:

$$Pr\left(w_n|w_{n-1}\right)=\frac{N\left(\mathrm{w}_i=w_{n-1}\,w_n\right)}{\sum_k N\left(w_{n-1}w_n^k\right)}, \quad (1)$$

где $Pr\left(w_n|w_{n-1}\right)$ — вероятность встретить слово w_n после слова w_{n-1}; N — счетчик числа встречающейся последовательности слов w_{n-1} и w_n в исследуемых текстах.

Векторная модель word2vec представляется в виде

$$Pr\left(w_n|w_c\right)=\frac{\mathbf{e}^{u_{w_c}^T V_{w_i}}}{\sum_C \mathbf{e}^{u_{w_c}^T V_{w_i}}}, \quad (2)$$

где Pr — вероятность встретить слово w_i рядом (до или после) с выбранным набором слов w_c (именуемым контекстом), являющимся базисом $[w_c]$ для представления слова w_i вектором $\mathbf{w}_i$; $V_{\mathbf{w}_i}$ — булевский вес вектора $\mathbf{w}_i$ для выбранного слова w_i, который принимает значение 0 или 1 в зависимости от отсутствия или наличия в тексте избранного слова; $u_{w_c}^T$ — вес w_c в тексте T.

Представленные выше модели (1) и (2) и другие им подобные базируются на исходном мультимножестве словосочетаний (словаре), составленном из выбранных для обучения наборов текстов (корпусе). Точность модели зависит от мощности словаря и распределения частот словосочетаний в коллекции. Подход к выбору исходных текстов, как правило, экспертно-субъективный, эмпирический, без учета временного фактора. Отсюда получаем перечисленные далее недостатки этого подхода.

1. Исходные тексты могут изначально не представлять реальной ценности для решения задач в интересующей предметной области в текущее время, что означает "знания" неактуальны.

2. Исходные тексты могут иметь высокую частоту обновления содержащихся в них моделей знаний, а следовательно, через незначительное время собранные "знания" станут неактуальными. Необходимым условием работы с такой моделью является возможность определения эффективного периода актуальности для каждого объекта модели знания (н-граммы или слова). Достаточным условием является прогноз изменения частоты (или иной качественной характеристики) встреч заданного объекта модели знаний в текстах будущего (выбранного) периода актуальности.

Для решения задач, устраняющих недостаток п. 1 на практике, чаще всего прибегают к экспертной оценке и последующему "обогащению" данных на основе больших данных, наращивая мощность словаря (как правило, более 1 000 000 слов). Однако такой подход не позволяет решать задачи, направленные на устранение недостатков п. 2, где требуется, наоборот, сокращение объема данных, подлежащих анализу. Решения таких задач в существующих моделях пока не представлены. Как следствие, необходим подход, позволяющий учитывать оценку изменения языковой модели представления знаний $\langle C_T, Pr \rangle$ (сформированной на основе словосочетаний w корпуса текстов T) для прогноза вероятности использования словосочетания w в текстах будущего периода.

Решение поставленной задачи возможно двумя путями: "от частного к целому" — оценкой динамики изменения каждого объекта и последующей интегральной оценкой степени изменения (пригодности) модели в будущем; дедуктивно — оценкой динамики изменения модели в целом и последующим учетом этого изменения для каждого словосочетания в будущем. Основываясь на опыте работы [3], в которой рассмотрена модель для отслеживания изменения мемов, как устойчивых "мутирующих" словосочетаний (*meme tracker*, стэндфордская модель), и работе [4], где описана компьютерная модель процесса эволюции культуры (*memes and variations*, MAV), был сделан вывод о целесообразности выбора эволюционного подхода "от частного к общему" при решении задачи оценки изменения и прогноза содержания ИП. Исходя из данного выбора, были сформулированы две подзадачи для решения целевой задачи оценки изменения языковой модели:

— определить количественную меру отличия одной языковой модели вида $\langle C_T, Pr(w) \rangle$ относительно объектов (словосочетания) от другой модели такого же вида;

— разработать алгоритм для прогноза относительной частоты P словосочетания w_i в следующий момент времени (через выбранный интервал времени) для языка, представленного языковой моделью вида $\langle C_T, Pr(w) \rangle$.

Решением поставленных задач явилось создание мем-грамм-модели (МГМ). Описание решения первой задачи будет дано в другой статье. В настоящей статье представлено решение второй из перечисленных выше подзадач в виде описания модели и алгоритма ее использования для прогноза числа копий мема за недельный интервал времени.

1. Базовые понятия мем-грамм-модели

Далее представлена разработанная авторами модель на основе форматизированного понятия мема, представленного биграммой (мем-граммы) в следующий за настоящим моментом времени прогнозируемый временной период (далее кратко — следующий период).

В н-грамм-модели используются следующие подходы к представлению и оценке моделей знаний, сформированных на базе текста [2]:

— текст рассматривается как цепь Маркова (в том числе высших порядков);

— оценкой качества модели является мера правдоподобия, перплексия или иные критерии.

Предлагаемая модель базируется на данных подходах со следующими, указанными далее дополнениями.

1. Формализуется понятие ИП.

На входе модели поступают мультимножества последовательностей н-грамм w — текстов $\{T_1^\tau, T_2^\tau, \ldots, T_i^\tau\}$ с присвоенной им меткой времени τ, число текстов T_i^τ конечно. За начальное время τ_0 выбирается значение времени самого раннего текста, определяется значение времени самого позднего текста $\tau_{\min}$, затем выбирается дискретный интервал Δt. Предлагается два подхода к выбору Δt. Первый из них — выбрать наиболее "эффективный" интервал, такой, чтобы можно было обнаружить большинство изменений в языковых моделях при минимальном числе равных интервалов. Второй — "предикативный" подход, при котором выбираемый интервал Δt дает лучшие условия (группировки н-грамм) для решения задачи прогноза. Первый вариант рассмотрен в работе [5], второй будет представлен в данной статье (в описании процесса разработки Алгоритма 1). После определения шага дискретизации Δt рассчитывается число целых интервалов $u = \dfrac{(\tau_{\min} - \tau_0)}{\Delta t}$, $u \in \mathbb{N}$, задаются дискретные интервалы времени $\{T_1^{t_1}, T_2^{t_2}, \ldots, T_i^{t_p}\}$, $t_p = \tau_0 - p\Delta t$, $n \in \mathbb{N}$, $0 < n \leqslant u$, и формируется мультимножество текстов с дискретной меткой времени $\{T_1^{t_1}, T_2^{t_2}, \ldots, T_i^{t_p}\}$. Из данного мультимножества текстов формируется ИП.

Определение 1. Информационным пространством (ИП) с метрикой силы информационного воздействия на наследственность содержания ИП будем называть тройку $\left(\left\langle C^{Pr}, M^{\varphi}\right\rangle, \alpha^{IS}\right)$, где C^{Pr} — коллекция языковых (н-грамм) моделей $\left\langle C_p, Pr(w_k)\right\rangle$ с единой функцией оценки условной вероятности $Pr\left(w_j^p = w_{n-1}w_n\right)$ события получения после слова w_{n-1} н-граммы $w_j^p = w_{n-1}w_n$ в произвольном тексте $T_i^{t_p}$ момента времени p; M^{φ} — коллекция множеств M_p единиц наследственности — мемов m_i^p — общих элементов мультимножеств C_{p-k+1} и C_{p-k}, сформированных посредством процедуры φ; $p > 1$; $k, p, i, j \in \mathbb{N}$, $C_p = \left\{w_1^p, w_2^p, ..., w_n^p\right\}$ — мультимножество н-грамм текстов $T_n^{t_p}$ одного момента времени p; α^{IS}-метрика, определяющее меру различия двух языковых моделей C_p относительно всех мемов одного из множеств M_p (правого — при рассмотрении в перспективе, левого — при рассмотрении в ретроспективе).

Отметим, что в рамках данной работы рассматриваются языковые н-грамм-модели (для случая биграмм), в которых вероятность мема, представленного биграммой, определяется как относительная частота встречи биграммы, являющейся мемом, относительно всех биграмм ИП. Подробно подходам к мере α^{IS} будет посвящена другая статья.

2. На ИП задается мем-грамм-модель. С применением моделей на н-граммах решается задача предсказания вероятности того, что определенная последовательность слов является допустимой для данного языка C. В мем-грамм-модели решается иная задача — предсказать, что допустимая для языка C_p последовательность слов $w_{n-1}w_n$ в текущий момент времени p будет допустимой для языка C_{p+1} в следующий момент времени $p + 1$. Это условие означает — быть мемом $m_i^{p+1} = w_{n-1}w_n$ для языков C_p и C_{p+1}. Формализированная запись данного высказывания:

$$Pr\left(m_j^{p+1} = w_i^p\right) = \frac{N\left(w_j^{p+1} = m_i^{p+1}\right)}{\sum_q N\left(w_q^{p+1} = m_i^{p+1}\right)}, \qquad (3)$$

где m_i^{p+1} — мем будущего периода; $w_i^p = w_{n-1}w_n$ — известная биграмма языковой модели C_p из коллекций языковых моделей $C^{Pr} : w_i^p \in C_p \subseteq C^{Pr}$.

Таким образом, для оценки (3) необходимо построение алгоритма Λ для прогноза числа $N\left(w_j^{p+1} = m_i^{p+1}\right)$ копий словосочетания w_j^{p+1}, являющегося мемом m_j^{p+1}. Учитывая, что на практике интересен не любой алгоритм, а только те, которые обладают заданным параметром качества, для $\exists\Lambda$ необходимо потребовать оценку θ достоверности прогноза интервала числа копий мема $N^{\Lambda}\left(w_j^{p+1} = m_i^{p+1}\right) \in [N_a; N_b]$. Длина указанного интервала должна быть гарантированно меньше заданного ξ:

$$\begin{cases} P\left(N_a \leqslant N^{\Lambda}\left(w_j^{p+1} = m_i^{p+1}\right) \leqslant N_b\right) \geqslant \theta; \\ N_b - N_a \leqslant \xi. \end{cases} \qquad (4)$$

Определение 2. Мем-грамм-модель задана, если задано ИП $\left(\left\langle C^{Pr}, M^{\varphi}\right\rangle, \alpha^{IS}\right)$ и определен $\Lambda^{\theta,\xi}$ — алгоритм (или функция) прогноза в следующий момент времени $p + 1$ числа $N\left(w_j^{p+1} = m_i^{p+1}\right)$ экземпляров словосочетания w_j^{p+1}, являющегося мемом $m_j^{p+1} \in M_{p+1} \not\subset M^{\varphi}$, относительно словосочинений $w_q^p \in C_p \subset C^{Pr}$ ИП текущего момента времени p; $p > 1$, $p, q, i, j \in \mathbb{N}$, θ — оценка достоверности даваемого алгоритмом прогноза; ξ — максимальная ошибка даваемого алгоритмом прогноза. В соответствии с данными дополнениями необходимым условием для задания мем-грамм-модели является задание четверки элементов $\left(\left\langle C^{Pr}, M^{\varphi}\right\rangle, \alpha^{IS}, \Lambda^{\theta,\xi}\right)$.

На настоящее время представляется сложным разработать математический подход к определению функции $\Lambda^{\theta,\xi}$. Трудности на этом направлении в значительной степени обусловлены необходимостью корректного обоснования ряда гипотез, основанных на наиболее детальном изучении процессов изменения реального ИП. Для этих целей сначала был разработан инструментарий (программное обеспечение) для сбора и обработки реальных данных — 3000 публикаций русскоязычного сегмента Интернета за июнь—август 2014 г. [6], а затем на их основе были созданы языковые н-грамм-модели, коллекция которых определяет объект изучения — ИП русскоязычного сегмента Интернета (ИПР) [5]. Изучение процессов, происходящих в ИПР, позволило сформировать метрику силы информационного воздействия и подтвердить целесообразность использования концепции меметики для описания изменений ИП. Результаты такого изучения позволили сформировать гипотезу "Об эволюции популяции мемов", которая впоследствии подтвердилась в виде доказанной теоремы "О плотности вероятности функции числа копий мемов", что в итоге привело к возможности создания алгоритма $\Lambda^{\theta,\xi}$ для расчета прогнозного числа копий мема. Основой для подтверждения гипотезы, доказательства теоремы и определения алгоритма является разработанная и представленая в настоящей статье статистическая модель для прогноза числа копий мема в следующий за настоящим период времени.

2. Постановка задачи

Целью исследования изменений ИП является разработка статистической имитационной модели ИП для прогноза числа копий мема в следующий период времени. Такая модель необходима для проверки гипотезы "Об эволюции популяции мемов". В случае ее подтверждения необходимо разработать алгоритм для применения модели на практике.

Исходным объектом анализа являются объекты ИП $\left(\left\langle C^{Pr}, M^{\varphi}, \alpha^{IS}\right\rangle\right)$, представленного коллекцией

C^{Pr} мультимножеств C_p словосочетаний $w_i^p = w_{n-1}w_n$ (биграмм), полученных из множества текстов публикации T_i^τ произвольной длины с датировкой времени τ, которое можно разделить на "до" и "после" выбранного момента времени τ_0. Биграмма — словосочетание двух слов, приведенных к лемматизированной форме. Например, биграмма "мама мыть" получена из словосочетания "мама мыла". Соответственно, из множества текстов, представленных последовательностью биграмм, формируется коллекция мультимножеств C^{Pr}, а из общих биграмм мультимножеств C_{p-k+1}, $C_{p-k} \in C^{Pr}$ за последовательные моменты времени $p - k + 1$ и $p - k$ формируется коллекция мемов M^φ.

В дальнейшем будем использовать следующие обозначения:

$t_p = \begin{cases} \tau_0 - p\Delta t, & p < k \\ \tau_0 + p\Delta t, & p \geqslant k \end{cases}$ — момент времени, где $t_p \in \{t_1, t_2, ..., t_k = \tau_0, ..., t_n\}$ или кратко — момент времени p;

$N\left(w_i^p\right)$ — число копий w_i биграммы в момент времени p;

$Sum\left(N_p\right) = \sum_i N\left(w_i^p\right)$ — суммарное число копий биграмм в момент времени p;

$S_T^{IS} = \dfrac{\sum_p Sum\left(N_p\right)}{p}$ — среднее число биграмм за период времени T, используется для нормировки характеристик биграмм ИП;

$Sum\left(M_p\right) = \sum_i N\left(w_i^p = m_i^p\right)$ — суммарное число копий мемов в момент времени t_p, определяет размер ИП;

$NN\left(w_i^p\right) = \dfrac{N\left(w_i^p\right)}{S_T^{IS}}$ нормированное число копий w_i биграммы в момент времени p;

$NSum\left(N_p\right) = \dfrac{Sum\left(N_p\right)}{S_T^{IS}} \cdot 10^6$ — нормированный размер ИП, определяемый суммарным числом копий всех биграмм в момент времени t_p. Нормирование проводится по размеру ИП в один миллион биграмм (принято для удобства использования, поскольку число копий исследуемого ИП, принятого за базовое, измеряется в миллионах биграмм).

Перечислим задачи исследования.

1. Провести первичный анализ имеющихся данных: оценить количественное и качественное изменение содержания ИП во времени, выявить оптимальный период для анализа, провести исследование распределения числа копий биграмм — формирование отклика исследования с помощью логарифма от числа копий биграмм.

2. Провести статистическое исследование влияния характеристик биграмм и ИП на число копий биграмм в следующий дискретный момент времени $p + 1$.

3. Провести регрессионное моделирование распределения случайной величины числа копий в следующий дискретный момент времени $p + 1$: выявить наиболее значимые факторы, построить регрессионную модель для логарифма числа копий биграмм в следующий дискретный момент времени $p + 1$.

4. Подобрать теоретическое распределение для аппроксимации распределения остатков построенной регрессионной модели для логарифма числа копий биграмм в следующий дискретный момент времени $p + 1$: проверка гипотезы о принадлежности эмпирического распределения основным теоретическим законам.

5. Теоретическое исследование распределения числа копий биграмм в следующий дискретный момент времени $p + 1$ (вывод основной Теоремы).

6. Валидация построенной статистической модели распределения числа копий биграмм в следующий дискретный момент времени $p + 1$ на реальных данных.

7. Разработка алгоритма построения прогноза распределения числа копий биграмм в следующий момент времени $p + 1$ на основе исторических данных за последние три периода: $p - 2$, $p - 1$, p.

3. Статистический анализ данных.

В работе [5] автор детально описал, каким образом были выбраны источники данных и как проводилась их обработка для формирования ИП. Отметим кратко наиболее важную для дальнейшего изложения информацию.

3.1. Подготовка данных для анализа

В целях изучения изменения русскоязычного сегмента ИП было отобрано порядка 780 000 публикаций СМИ с 01.05.2014 по 01.04.2015, ранжируемых сервисом Meediametics.ru как наиболее цитируемые в социальных медиа. Публикации были обработаны методом, представленным в работе [5], в данные в виде массива строк: биграмма; число ее копий в указанную дату; дата (с шагом 1 день). Общее число биграмм (оригиналов по дням) в исходных данных составило 1 731 496.

Анализ исходных данных показал, что изменения в ИП имеют выраженную недельную сезонную составляющую. На рис. 1 приведены распределения по числу копий биграмм для каждого из дней недели. Как видно на графике, число копий биграмм постепенно нарастает с понедельника по пятницу и снижается в выходные дни.

Для снятия эффекта "влияния дня недели" целесообразно перейти от ежедневных данных к данным по неделям. При этом для построения модели будем использовать предысторию за два предыдущих периода.

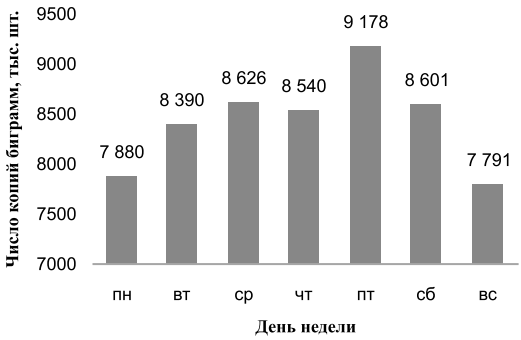

Рис. 1. Гистограмма распределения числа копий биграмм по дням недели

В целях наглядного представления процессов изменения состава ИП приведем пример количественных характеристик десяти наиболее часто встречающихся мемов, представленных в табл. 1.

3.2. Группировка данных

Статистический анализ количественных характеристик биграмм показал, что имеет место высокая корреляционная зависимость числа копий мема $N\left(w_i^{p+1}\right)$ от динамики изменения в предыдущие три момента времени (табл. 2). Исходя из определения мема, для утверждения, что мем будет завтра, необходимо и достаточно появление хотя бы одной его копии. В соответствии с постановкой задачи, рассмотрим вероятность появления мема $P\left(N\left(w_i^{p+1}\right)\right)$ в зависимости от числа упоминаний этого мема в последовательные два предыдущих и текущий моменты времени. Тогда имеет место функциональная зависимость f так, что

$$P\left(N\left(w_i^{p+1}\right)\middle| N\left(w_i^{p-2}\right),\ N\left(w_i^{p-1}\right),\ N\left(w_i^{p}\right)\right) =$$
$$= f\left(N\left(w_i^{p-2}\right),\ N\left(w_i^{p-1}\right),\ N\left(w_i^{p}\right)\right).$$

На основе анализа исходных данных методом дерева решений были получены результаты, представленные

Таблица 1

Первые десять наиболее популярных мемов

№	Мем	Общее число упоминаний	Среднее число упоминаний в день
1	"об это"	41 261	156
2	"риа новость"*	36 270	137
3	"владимир путин"	30 932	117
4	"то число"	29 200	111
5	"кроме то"	22 808	86
6	"который быть"	19 461	73
7	"человек который"	17 336	65
8	"б б"	16 616	102
9	"самый дело"	16 034	61
10	"народный республика"	15 273	58

*Технический мем, вызванный ошибкой обработки публикации, связан с указанием источника в публикации, а не с ее содержанием.

Таблица 2

Группы мемов

Группа мемов	Число копий мема в момент времени			Вероятность появления мема в период $(p+1)$	Число записей* (объем выборки)
	$p-2$	$p-1$	p		
1	$N\left(w_i^{p-2}\right) \geqslant 1$	$N\left(w_i^{p-1}\right) \geqslant 2$	$N\left(w_i^{p}\right) \geqslant 4$	0,859	457 049
2	$N\left(w_i^{p-2}\right) \geqslant 1$	$N\left(w_i^{p-1}\right) \geqslant 2$	$N\left(w_i^{p}\right) < 4$	0,596	46 894
3	$N\left(w_i^{p-2}\right) \geqslant 1$	$N\left(w_i^{p-1}\right) < 2$	$N\left(w_i^{p}\right) \geqslant 4$	0,663	372 870
4	$N\left(w_i^{p-2}\right) \geqslant 1$	$N\left(w_i^{p-1}\right) < 2$	$N\left(w_i^{p}\right) < 4$	0,425	143 290
5	$N\left(w_i^{p-2}\right) = 0$	$N\left(w_i^{p-1}\right) \geqslant 2$	$N\left(w_i^{p}\right) \geqslant 4$	0,507	134 646
6	$N\left(w_i^{p-2}\right) = 0$	$N\left(w_i^{p-1}\right) \geqslant 2$	$N\left(w_i^{p}\right) < 4$	0,271	38 146
7	$N\left(w_i^{p-2}\right) = 0$	$N\left(w_i^{p-1}\right) < 2$	$N\left(w_i^{p}\right) \geqslant 4$	0,417	554 664
8	$N\left(w_i^{p-2}\right) = 0$	$N\left(w_i^{p-1}\right) < 2$	$N\left(w_i^{p}\right) < 4$	0,110	557 462

* Запись – уникальная комбинация идентификатора мема и номера недели.

в табл. 2. Данные результаты содержат оценки вероятности появления мема в зависимости от числа упоминаний за три предыдущих периода времени. Отметим, что под вероятностью появления мема понимается доля событий $N\left(w_i^{p+1}\right) > 0$ относительно суммы событий $N\left(w_i^{p+1}\right) > 0$ и $N\left(w_i^{p+1}\right) = 0$.

Таким образом, удалось выделить восемь групп исходных данных, отражающих различные предположения относительно $N\left(w_i^{p-2}\right)$, $N\left(w_i^{p-1}\right)$ и $N\left(w_i^{p}\right)$. Наиболее вероятное появление мема в следующем периоде получаем в группе 1 при условии, что $N\left(w_i^{p-2}\right) \geqslant 1$, $N\left(w_i^{p-2}\right) \geqslant 2$ и $N\left(w_i^{p}\right) \geqslant 4$, вероятность дальнейшего появления мема равна

$$P\left(N\left(w_i^{p+1}\right) > 0 \middle| N\left(w_i^{p-2}\right),\ N\left(w_i^{p-1}\right),\ N\left(w_i^{p}\right)\right) = 0{,}859.$$

Поскольку группа 1 — единственная группа, в которой у мема имеется полноценная история за последние три периода, то эту группу данных будем в дальнейшем использовать в качестве исходных данных для регрессионного моделирования. Выбор обусловлен тем, что мемы групп 5—8 не имеют историю за три периода, а это минимальное условие характеристики мема в поставленной задаче. Мемы групп 2—4 не имеют устойчивого роста, и если будут "успешны", то только когда перейдут в группу 1. Такой подход позволяет рационально использовать вычислительные и временные ресурсы для мемов с незначительным числом копий (группы 2—8), поскольку позволяется упростить задачу прогнозирования за счет снижения требования к точности прогноза: с "узкого" интервала числа копий н-граммы на "широкий" интервал — от одной и более копий н-граммы.

3.3. Регрессионное моделирование

Построение регрессионной модели будем проводить для совокупности данных, соответствующих группе 1 из табл. 2. Следует отметить, что для описания положительных случайных величин, аналогичных рассматриваемому в статье числу копий мема в момент времени (p + 1), чаще всего используют распределения из семейства экспоненциальных [7, 8]. В этом случае применяют мультипликативно-экспоненциальные регрессионные уравнения следующего вида:

$$y = \mathbf{e}^{\mathbf{BX}}\varepsilon, \qquad (5)$$

где y — отклик, $\mathbf{B}$ — вектор неизвестных параметров; $\mathbf{X}$ — вектор входных факторов; ε — случайная ошибка.

При логарифмировании левой и правой частей уравнения (5) получаем аддитивную множественную линейную регрессионную модель $\ln y = \mathbf{BX} + \ln\varepsilon$, параметры которой можно оценивать любым из классических методов, например, методом наименьших квадратов. В связи с этими соображениями в дальнейшем в качестве отклика регрессионной модели будем рассматривать величину $\ln\left(NN\left(w_i^{p+1}\right)\right)$. Для построения модели будем использовать шаговые алгоритмы регрессионного анализа [9]. В качестве регрессоров рассматривались функции числа копий мема в следующий период времени: от основных характеристик мема; от мультипликативных характеристик мема; от характеристик информационного пространства.

В ходе дальнейшего анализа исходные данные были разделены на обучающую и контрольную выборки. Обучающую выборку составили данные, полученные с 21-й по 30-ю недели, контрольную — данные, полученные с 31-й по 34-ю недели.

Все приведенные возможные регрессоры проверяли на корреляционную зависимость с откликом. Для этого применяли пакет прикладных программ для статистического анализа JMP SAS: проводили расчет коэффициентов корреляций и их значимости. В качестве корреляций использовали коэффициенты Пирсона и Спирмена. В результате проверки было выявлено, что все факторы (регрессоры), в той или иной мере характеризующие размерность ИП, не оказывают статистически значимого влияния на $\ln\left(NN\left(w_i^{p+1}\right)\right)$. Статистически значимые коэффициенты корреляции по обучающей и кон-

Таблица 3

Топ 5 регрессоров, оказывающих статистически значимое влияние на отклик

Возможный регрессор	Коэффициент корреляции с $\ln\left(NN\left(w_i^{p+1}\right)\right)$	
	Обучающая выборка	Контрольная выборка
$\ln\left(NN\left(w_i^{p}\right)\right)$	0,6692	0,6641
$\ln\left(NN\left(w_i^{p-1}\right)\right)$	0,6209	0,6248
$\ln\left(NN\left(w_i^{p-2}\right)\right)$	0,6116	0,5898
$NN\left(w_i^{p}\right)NN\left(w_i^{p-1}\right)$	0,1485	0,1605
$NN\left(w_i^{p-1}\right)NN\left(w_i^{p-2}\right)$	0,1434	0,1460

Таблица 4

Показатели качества модели

Показатель	Обучающая выборка	Контрольная выборка
R^2	0,527452	0,516364
Средняя квадратическая ошибка	0,639258	0,648981
Число наблюдений	292 307	100 181

Таблица 5

Статистические характеристики оценок параметров

Регрессор	Оценки параметров*	Стандартная ошибка	*t*-статистика	*P*-значения	95 %-ный доверительный интервал	
					Нижняя граница	Верхняя граница
a_0	−0,069779	0,003311	−21,07	<0,0001	−0,07627	−0,06329
$\ln\left(NN\left(w_i^p\right)\right)$	0,493715	0,002325	212,35	<0,0001	0,489158	0,498272
$\ln\left(NN\left(w_i^{p-1}\right)\right)$	0,2133316	0,002065	103,31	<0,0001	0,209284	0,217379
$\ln\left(NN\left(w_i^{p-2}\right)\right)$	0,2366009	0,00179	132,2	<0,0001	0,233093	0,240109
* Оценки получены на основе метода наименьших квадратов.						

трольной выборкам были получены только для десяти из двенадцати рассмотренных регрессоров. Пять из наиболее значимых регрессоров приведенны в табл. 3.

Количество моделей, которые теоретически могли бы быть построены с использованием даже 12 регрессоров, очень велико и составляет более чем $\sum_{k=1}^{12} C_{12}^k = 2^k - 1 = 4095$ единиц. Задача выбора наилучшей модели решалась с использованием специальных итерационных шаговых алгоритмов [9], реализованных в большинстве современных статистических программных пакетов. Из всего множества рассмотренных моделей (с различными регрессорами) наиболее предпочтительной (по значению R^2) оказалась авторегрессионная модель с лаговыми переменными (значение которой взято за предыдущие моменты времени) от периодов времени p, $(p-1)$ и $(p-2)$ следующего вида:

$$\ln\left(NN\left(w_i^{p+1}\right)\right) = a_0 + a_1 \ln\left(NN\left(w_i^p\right)\right) + \\ + a_2 \ln\left(NN\left(w_i^{p-1}\right)\right) + a_3 \ln\left(NN\left(w_i^{p-2}\right)\right) + \varepsilon. \quad (6)$$

Таблица 6

Перцентильная группировка наблюдений по значению регрессии

Перцентили	Объем подвыборки		Среднее значение регрессии	
	Обучения	Контроль	Обучения	Контроль
0,005	2024	768	0,62828	0,62800
0,025	5807	1927	0,72407	0,72539
0,05	6955	2293	0,79669	0,79507
0,1	14 162	4844	0,87986	0,87881
0,2	29 517	9881	0,99956	0,99985
0,3	29 174	9849	1,13747	1,13615
0,4	29 343	10 048	1,26412	1,26446
0,5	29 211	10 091	1,39745	1,39664
0,6	29 192	9816	1,54322	1,54306
0,7	29 229	10 560	1,71909	1,71600
0,8	29 232	10 237	1,94535	1,94405
0,9	29 230	9761	2,28319	2,28281
0,95	14 616	4942	2,70246	2,69844
0,975	7308	2448	3,11692	3,12248
0,995	5846	2207	3,69460	3,70338
1	1461	509	4,71040	4,65646

Результаты оценивания параметров модели (6) и оценка ее качества на обучающей и контрольной выборках приведены в табл. 4 и 5. Как видно из данных табл. 4, на обучающей и контрольной выборках с помощью модели удается охватить более 50 % вариаций отклика (значения $R^2 > 0,5$).

Отметим, что все параметры модели обладают хорошей значимостью (табл. 6).

С использованием данной модели прогнозируемая величина $\ln\left(Sn_i(t+1)\right)$ может быть получена по следующей формуле:

$$\ln\left(NN\left(w_i^{p+1}\right)\right)_{\text{Прогноз}} = -0,069779 + 0,493715 \times \\ \times \ln\left(NN\left(w_i^p\right)\right) + 0,2133316 \ln\left(NN\left(w_i^{p-1}\right)\right) + \\ + 0,2366009 \ln\left(NN\left(w_i^{p-2}\right)\right). \quad (7)$$

На рис. 2 представлены графики рассеивания наблюдаемых значений переменной $\ln\left(NN\left(w_i^{p+1}\right)\right)$

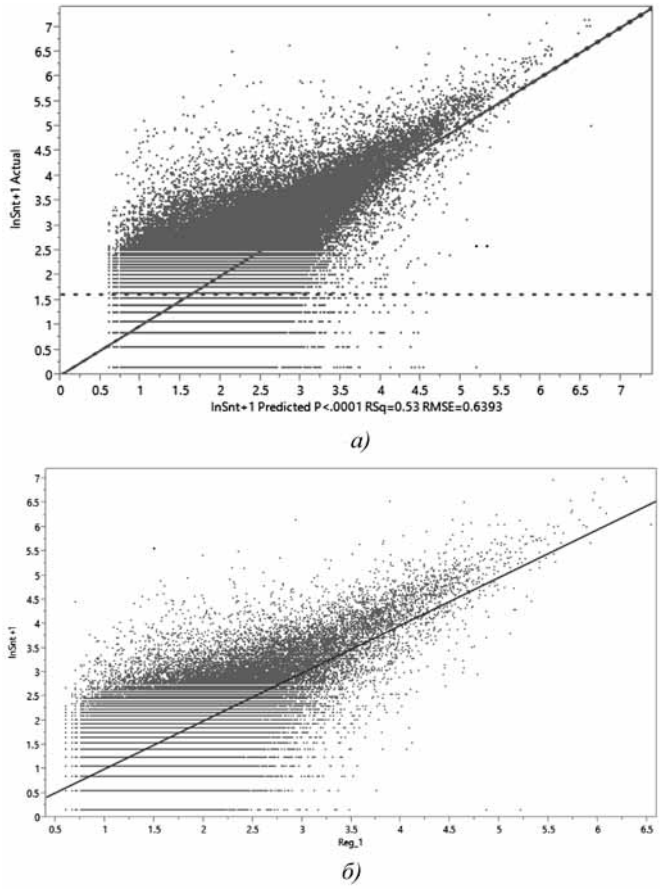

Рис. 2. Диаграмма рассеивания наблюдаемых значений в зависимости от прогнозированных значений для переменной $\ln\left(NN\left(w_i^{p+1}\right)\right)$:
а — по обучающей выборке; *б* — по контрольной выборке

в зависимости от значений прогнозируемых величин соответственно для обучающей и контрольной выборок. Как видно на рис. 2, в обоих случаях наблюдается ярко выраженная неоднородность исходных данных, а именно, при уменьшении значений переменных вариация данных существенно возрастает.

Оценки параметров, представленных ранее в табл. 4 и 5, были получены с использованием обычного метода наименьших квадратов [9], который не учитывает возможную неоднородность исходных данных. Для корректного оценивания параметров необходимо получить более детальную информацию о распределении отклика. Для решения этой задачи была проведена группировка значений регрессии по децентилям согласно табл. 6.

Такая группировка дает возможность сформировать однородные группы, в которых наблюдается одинаковое распределение отклика, независящее от регрессоров. В каждой такой группе корректно провести процедуру оценки параметров уже не для остатков (разницы фактического и прогнозируемого значений), а для исходных значений отклика. Таким способом устраняется влияние значения регрессоров.

3.4. Подбор распределения для отклика

Для подбора подходящего распределения случайной величины $\ln\left(NN\left(w_i^{p+1}\right)\right)$ выделим сегмент, имеющий наиболее ярко выраженную взаимосвязь межу регрессией (7) и $\ln\left(NN\left(w_i^{p+1}\right)\right)$. Этот сегмент соответствует значению перцентиля от 0,995 до 1 в табл. 6, правой верхней части облака рассеивания точек на рис. 2. Процесс оценивания формы распределения будем проводить по остаткам регрессии для сегмента перцентиля от 0,995 до 1, используя обучающую и контрольную выборки. Целесообразность такого выбора объясняется тем обстоятельством, что на практике представляет интерес выявления мема следующего периода не на самых ранних стадиях зарождения (2—3 копии), когда вероятность ложного детектирования весьма велика, а на стадии начального роста (10...20 копий), где вероятность ошибки меньше. Такой стадии и соответствует сегмент перцентиля от 0,995 до 1. Цель данного анализа — подобрать наилучшее распределение для случайной величины. Зная закон распределения остатков, будем знать закон распределения для отклика (так как они будут совпадать). В качестве возможных распределений будем рассматривать наиболее известные распределения, используемые в математической статистике для описания распределений регрессионных остатков: нормальное, мультинормальное, Лапласа, Коши, логистическое, минимальных значений, L-распределение и распределение Su-Джонсона [8, 9].

Для каждого из перечисленных распределений была проведена проверка статистической гипотезы о принадлежности выборки указанному закону распределения, т. е. проверки того, что эмпирическое распределение соответствует предполагаемой модели. Проверка осуществлялась с использованием критерия согласия Колмогорова [8, 9].

В табл. 7 приведены результаты проверки согласия имеющихся данных по остаткам модели с перечисленными выше распределениями, а также полученные по выборке параметры функций распределений. Для дальнейшей работы было выбрано распределение Su-Джонсона, так как ему соответствует минимальное (0,58) значение статистики Колмогорова. Таким образом, на основании критерия согласия Колмогорова была принята гипотеза, что плотности распределения логарифма числа копий мемов в ИП описываются распределением Su-Джонсона.

Таблица 7

Результат проверки согласия

Распределение	Статистика Колмогорова	Параметры распределения*			
		Масштаб	Сдвиг	Асимметрия	Эксцесс
Лапласа	2,28	0,3251	0,2524	—	—
Нормальное	6,54	0,5979	0,1755	—	—
Коши	2,64	0,1944	0,2614	—	—
Логистическое	2,38	0,2272	0,242	—	—
Минимальных значений	3,51	0,4126	0,4014	—	—
L-распределение	1,55	0,1819	0,5701	0,5164	1,5936
Su-Джонсона	0,58	0,261	0,4047	0,4618	0,9661

* Оценку проводили на основе метода минимального расстояния "Омега-большое квадрат Мизеса". Данный метод обладает большей устойчивостью, нежели классический метод максимального правдоподобия [8, 9].
Примечание. Прочерк "—" означает недопустимость данных параметров для соответствующих распределений.

4. Модель на основе распределения Su-Джонсона

Применение распределения Su-Джонсона для моделирования процесса изменения числа копии мема невозможно без решения двух задач. Первая задача — разработать и доказать корректность преобразования плотности распределения Su-Джонсона логарифма случайной величины в плотность распределения случайной величины. Вторая задача — определить зависимость изменения дополнительных параметров распределения Su-Джонсона — масштаба, эксцесса, асимметрии и сдвига от логарифма числа копий.

Представим краткое описание предлагаемых подходов для решения этих задач и полученных результатов их применения.

4.1. Распределение Su-Джонсона применительно к результатам статистического анализа

Обобщим полученные результаты в виде теоретических выкладок. Для этого рассмотрим случайную величину ξ, распределенную по закону Su-Джонсона с плотностью распределения [8, 9]

$$f(x,\gamma,\delta,\lambda,\mu)=\frac{\delta}{\lambda\sqrt{2\pi}}\frac{1}{\sqrt{\left(\frac{x-\mu}{\lambda}\right)^2+1}}\times \\ \times\exp\left\{-\frac{1}{2}\left[\gamma+\delta\ln\left(\frac{x-\mu}{\lambda}+\sqrt{\left(\frac{x-\mu}{\lambda}\right)^2+1}\right)\right]^2\right\}, \quad (8)$$

где $\delta > 0$ — параметр эксцесса; γ — параметр асимметрии; $\lambda > 0$ — параметр масштаба; μ — параметр сдвига; $y \in (-\infty;+\infty)$.

Основные характеристики распределения (4):

Математическое ожидание

$$E\xi=\mu-\lambda\exp\left(\frac{\delta^{-2}}{2}\right)\sinh\left(\frac{\gamma}{\delta}\right).$$

Дисперсия

$$D\xi=\frac{\lambda^2}{2}\left(\mathbf{e}^{\delta^{-2}}-1\right)\left(\mathbf{e}^{\delta^{-2}}\cosh\left(\frac{2\gamma}{\delta}\right)+1\right). \quad (9)$$

Теорема 1. О плотности распределения композиции функции распределения Su-Джонсона и экспоненциальной функции (о плотности распределения числа копий мемов).

Пусть величина ξ имеет плотность распределения $f_\xi(x,\delta,\gamma)$ (8), тогда случайная величина $\eta = \mathbf{e}^\xi$ имеет плотность распределения

$$g(x,\gamma,\delta,\lambda,\mu)=\frac{1}{x}\frac{\delta}{\lambda\sqrt{2\pi}}\frac{1}{\sqrt{\left(\frac{\ln x-\mu}{\lambda}\right)^2+1}}\times \\ \times\exp\left\{-\frac{1}{2}\left[\gamma+\delta\ln\left(\frac{\ln x-\mu}{\lambda}+\sqrt{\left(\frac{\ln x-\mu}{\lambda}\right)^2+1}\right)\right]^2\right\}, \quad (10)$$

где $\delta > 0$ — параметр эксцесса; γ — параметр асимметрии; $\lambda > 0$ — параметр масштаба; μ — параметр сдвига; $y \in (-\infty;+\infty)$.

Доказательство. Сформулируем в виде вспомогательной теоремы известное из теории вероятностей утверждение о преобразовании плотности функции распределения с помощью монотонной функции [8]: пусть ξ имеет плотность распределения $f_\xi(x)$ (8) и функция $h(x)$ — монотонна, тогда случайная величина $\eta = \eta(\xi)$ имеет плотность распределения

$$f_\eta(x)=\left(h^{-1}(x)\right)'f_\xi\left(h^{-1}(x)\right).$$

Функция $h(x) = \mathbf{e}^x$ монотонна на всем промежутке значений x — условия вспомогательной теоремы удовлетворены, обратная функция для $h(x) = \mathbf{e}^x$: $h^{-1}(x) = \ln(x)$. Производная от $\left(h^{-1}(x)\right)' = \frac{1}{x}$. Для формулы (8) с учетом вспомогательной теоремы получаем исходное распределение (10).

Смысл Теоремы 1 раскрывается в возможность перехода от функции логарифма числа копий мема к функции от числа копий мема: получения оценки параметров распределения случайной величины $\ln\left(NN\left(w_i^{p+1}\right)\right)$ (8) для построения доверительных интервалов распределения случайной величины $NN\left(w_i^{p+1}\right)$ (10).

На рис. 3 представлены частные случаи распределений случайных величин ξ (8) и $\eta = \mathbf{e}^{\xi}$ (10) при нулевом сдвиге и единичном масштабе ($\mu = 0$; $\lambda = 1$).

4.2. Исследование параметров распределения Su-Джонсона

Изучим влияние значений регрессии на параметры распределения. Для этой цели проведем оценивание параметров распределения Su-Джонсона для отклика $\ln\left(NN\left(w_i^{p+1}\right)\right)$, построенных для каждой перцентильной группы от 0,005 до 1 на основе регрессионной модели (7). Оценку проводили на основе метода минимального расстояния "Омега-большое квадрат Мизеса" с применением программы для статистического анализа данных ISW 4.4.1.98 профессора Б. Ю. Лемешко[1].

В частности, для описания параметра асимметрии ***A*** подходят две модели: первые десять групп с коэффициентом детерминации (R^2) 0,9885 описываются логарифмической кривой, а остальные группы с коэффициентом детерминации 0,9725 — линейной функцией

$$A = \begin{cases} 4,368\ln(x) - 0,8087, & x < 1,719 \\ -0,4685x + 2,2014, & x \geqslant 1,719. \end{cases} \quad (11)$$

Для описания параметра эксцесса ***E*** подходят две модели: первые десять групп с коэффициентом детерминации 0,99325 описываются квадратичной параболой, а остальные группы с коэффициентом детерминации 0,9947 — линейной функцией

$$E = \begin{cases} 3,0039x^2 - 9,3989x + 9,9661, & x < 1,719 \\ -0,5393x + 3,5559, & x \geqslant 1,719. \end{cases} \quad (12)$$

Для описания параметра масштаба ***M*** можно использовать одну модель: с коэффициентом детерминации 0,999 эта зависимость логарифмической кривой

$$M = -1,112\ln(x) + 2,1283. \quad (13)$$

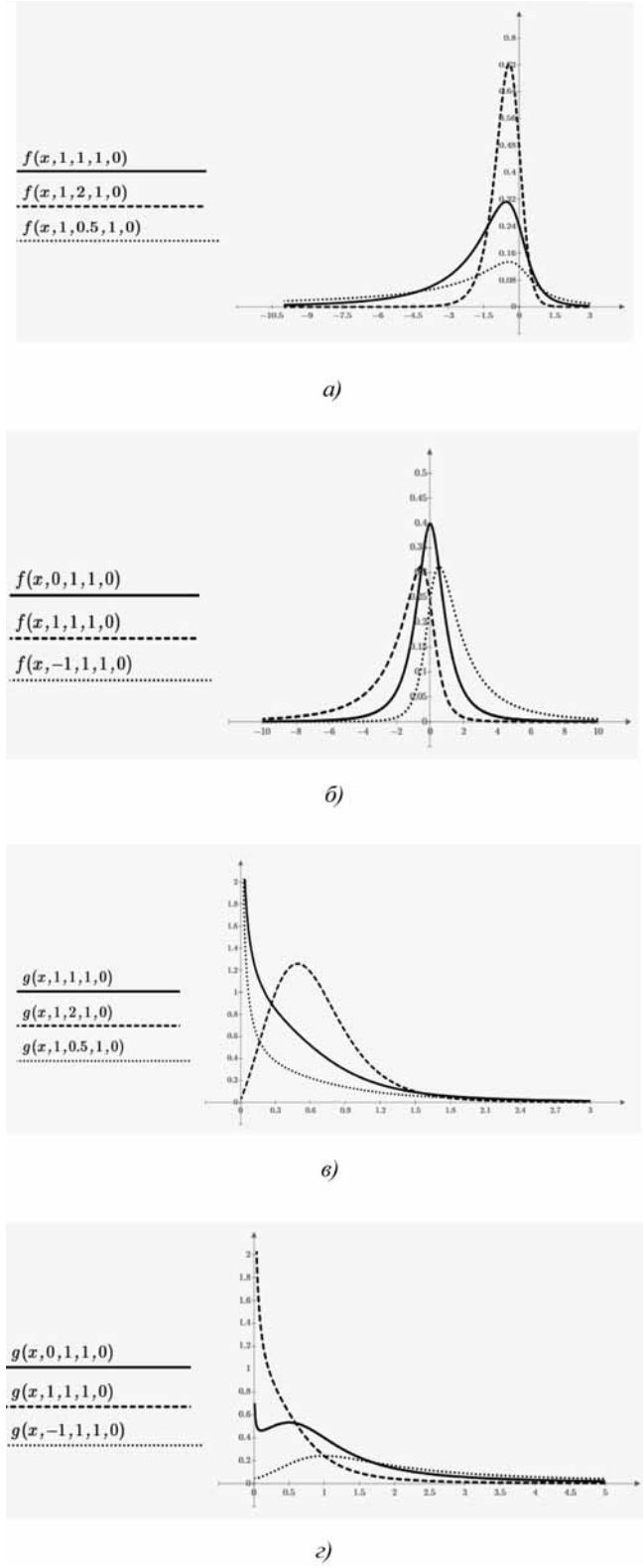

Рис. 3. Графики изменения плотности распределения Su-Джонсона в зависимости от эксцесса δ (*а*); асимметрии γ (*б*); масштаба (*в*) и сдвига λ (*г*)

[1] Программа доступна на сайте https://www.ami.nstu.ru/~headrd/

Для описания параметра сдвига S подходят две модели: первые десять групп с коэффициентом детерминации 0,9874 описываются первой линейной моделью, а остальные группы с коэффициентом детерминации 0,9947 — второй линейной моделью

$$S = \begin{cases} 3{,}1139x - 2{,}6093, & x < 1{,}719 \\ 0{,}7942x + 1{,}1385, & x \geqslant 1{,}719. \end{cases} \quad (14)$$

4.3. Валидация построенной регрессионной модели

Для проверки корректности построенной модели (7) было проведено ее тестирование на обучающей (число записей 292 307) и контрольной (число записей 100 181) выборках при условии, что все параметры модели (11)—(14) были оценены на основе обучающей выборки. В качестве оценки адекватности модели будем рассматривать качество построенных доверительных интервалов. В табл. 8 и 9 для сравнения приведены результаты тестирования при аппроксимации нормальным распределением и распределением Su-Джонсона.

Как видно из данных табл. 8, 9, средний модуль ошибки на обучающей и контрольной выборке при аппроксимации нормальным распределением составляет соответственно 1,9 и 1,6 %. При аппроксимации распределением Su-Джонсона средний модуль ошибки на обучающей и контрольной выборках существенно ниже и составляет соответственно 1,0 и 0,8 %.

5. Алгоритм прогноза числа копий мема

Прогноз количества копий мема в следующий момент времени для произвольного ИП, удовлетворяющий закон сохранения разнообразия, может быть осуществлен с использованием разработанной имитационной модели эволюции ИП по алгоритму, приведенному ниже.

Алгоритм прогноза числа копий мема в ИП следующего момента времени

1. Вход: S_T^{IS}, $N\left(w_i^p\right)$, $N\left(w_i^{p-1}\right)$, $N\left(w_i^{p-2}\right)$, $Sum\left(N_p\right)$ // размер эталонного ИП, число копий мема за три интервала времени, размер реального ИП.

1.1. $k = S_T^{IS} \cdot 10^{-6}$ // определение нормировочного коэффициента.

1.2. $NN\left(w_i^p\right) = \dfrac{N\left(w_i^p\right)}{k}$, $NN\left(w_i^{p-1}\right) = \dfrac{N\left(w_i^{p-1}\right)}{k}$, $NN\left(w_i^{p-2}\right) = \dfrac{N\left(w_i^{p-2}\right)}{k}$ // нормировка числа копий мема.

Таблица 8

Процент попадания значения $\ln\left(NN\left(w_i^{p+1}\right)\right)$ в доверительный интервал при аппроксимации нормальным распределением

Выборка	Доверительный интервал, %						
	95,0	90,0	75,0	50,0	25,0	10,0	5,0
Обучающая	93,0	87,9	75,5	52,8	23,6	7,5	3,4
Контрольная	92,6	87,1	74,4	52,3	24,7	8,3	3,9
Ошибочное попадание в доверительный интервал, %							
Обучающая	−2,0	−2,1	0,5	2,8	−1,4	−2,5	−1,6
Контрольная	−2,4	−2,9	−0,6	2,3	−0,3	−1,7	−1,1

Таблица 9

Процент попадания значения $\ln\left(NN\left(w_i^{p+1}\right)\right)$ в доверительный интервал при аппроксимации распределением Su-Джонсона

Выборка	Доверительный интервал, %						
	95,0	90,0	75,0	50,0	25,0	10,0	5,0
Обучающая	96,4	91,3	76,4	51,8	25,3	9,4	4,6
Контрольная	96,0	90,5	75,3	51,6	26,5	10,5	5,3
Ошибочное попадания в доверительный интервал, %							
Обучающая	−2,0	−2,1	0,5	1,8	0,3	−0,6	−0,4
Контрольная	1,0	0,5	0,3	1,6	1,5	0,5	0,3

2. Case: $\left(\left\{NN\left(w_i^p\right), \left(w_i^{p-1}\right), \left(w_i^{p-2}\right)\right\} \Rightarrow GroupID\left(w_i^{p+1}\right) = [1...8]\right)$ // определение группы мемов.

3. If $GroupID\left(w_i^{p+1}\right) \neq 1$ **Then**.

4. Выход 1: $N\left(w_i^{p+1}\right) = 1; \quad P\left(w_i^{p+1}\right) = P(GroupID)$ определяется вероятность, что будет хоть одна копия мема в следующий интервал времени.

5. Else

5.1. $x = -0,069779 + 0,493715\ln\left(NN\left(w_i^p\right)\right) + 0,2133316\ln\left(NN\left(w_i^{p-1}\right)\right) + 0,2366009\ln\left(NN\left(w_i^{p-1}\right)\right)$ // прогноз логарифма числа копий.

5.2. $$\gamma = \begin{cases} 3,1139x - 2,6093, & x < 1,719 \\ 0,7942x + 1,1385, & x \geqslant 1,719. \end{cases}$$

5.3. $$\delta = \begin{cases} 4,368\ln(x) - 0,8087, & x < 1,719 \\ -0,4685x + 2,2014, & x \geqslant 1,719. \end{cases}$$

5.4. $$\lambda = \begin{cases} 3,0039x^2 - 9,3989x + 9,9661, & x < 1,719 \\ -0,5393x + 3,5559, & x \geqslant 1,719. \end{cases}$$

5.5. $\mu = -1,112\ln(x) + 2,1283.$

5.6. $$x1 = \int_0^{1000} \frac{1}{x}\frac{\delta}{\lambda\sqrt{2\pi}}\frac{1}{\sqrt{\left(\frac{\ln x - \mu}{\lambda}\right)^2 + 1}} \times \exp\left(-\frac{1}{2}\left(\gamma + \delta\ln\left(\frac{\ln x - \mu}{\lambda} + \sqrt{\left(\frac{\ln x - \mu}{\lambda}\right)^2 + 1}\right)\right)^2\right) \times dx = 0,05.$$

5.7. $$x2 = \int_0^{1000} \frac{1}{x}\frac{\delta}{\lambda\sqrt{2\pi}}\frac{1}{\sqrt{\left(\frac{\ln x - \mu}{\lambda}\right)^2 + 1}} \times \exp\left(-\frac{1}{2}\left(\gamma + \delta\ln\left(\frac{\ln x - \mu}{\lambda} + \sqrt{\left(\frac{\ln x - \mu}{\lambda}\right)^2 + 1}\right)\right)^2\right) \times dx = 0,05.$$

5.8. **Выход 2:** $P\left(x1k \leqslant N\left(w_i^{p+1}\right) \leqslant x2k\right) = 0,9$ // определение интервала числа копий мема с 90 %-ной вероятностью.

6. **End**.

Отметим, что результат работы алгоритма — это возможность получения прогноза числа копий мема в диапазоне единиц и десятков единиц. Качество данного результата можно считать хорошим, поскольку возможно детектировать мемы на ранних этапах их зарождения. Наглядное сравнение данной оценки можно провести по количественным показателям "зрелых" мемов (см. табл. 1), насчитывающих от сотен до тысяч копий в неделю.

Заключение

Полученные в данном исследовании результаты кратко можно сформулировать в виде следующих положений.

- На основе анализа динамики изменения числа копий 200 000 биграмм, являющихся мемами (обучающая выборка), была разработана математическая модель, решающая задачу прогноза числа копий мема в ИП в следующий недельный интервал времени в зависимости от изменения числа копий данного мема за три предыдущих недельных интервала.
- Сформулирована и доказана теорема о распределении числа копий мема в следующий дискретный период времени $p + 1$.
- Валидация построенной статистической модели распределения числа копий мема в следующий дискретный момент времени на реальных данных показала высокую точность попадания реальных значений отклика в заданные 95 %-ные доверительные интервалы со средней погрешностью не более 1 %. Оценка качества разработанного модели (алгоритма), получена в результате проведения 100 000 экспериментов по прогнозу числа копий мема из контрольной выборки.

Авторы выражают благодарность научным руководителям д-ру физ.-мат. наук, проф. МГУ имени М. В. Ломоносова Валерию Александровичу Васенину и д-ру техн. наук, проф. Сергею Павловичу Расторгуеву за помощь в разработке мем-грамм-модели.

Список литературы

1. **Ramisch C.** N-gram models for language detection. URL: http://ru.scribd.com/doc/244210124/N-gram-Models-for-Language-Detection#

2. **Mikolov T., Chen K., Corrado G., Dean J.** Efficient estimation of world representation in vector space. URL: http://arxiv.org/pdf/1301.3781.pdf

3. **Leskovec J., Backstrom L., Kleinberg J.** Meme-tracking and the dynamics of the news cycle //Proceedings of the 15th ACM SIGKDD — International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2009. P. 497—506.

4. **Gabora L.** Meme and variations: A computational model of cultural evolution // 1993 Lectures in complex systems. Addison Wesley, 1995. P. 471—485.

5. **Артёмов А. А.** Эксперимент по моделированию процесса эволюции содержания информационного пространства социума (с применением мем-грамм-модели) // Программная инженерия. 2016. Т. 7, № 7. С. 291—306.

6. **Артёмов А. А.** Модель оценки уровня угроз информационных вызовов плану содержания информационного пространства социально-телекоммуникационной системы // Информационные войны. 2015. № 3 С. 83—97.

7. **Шуленин В. П.** Математическая статистика. Ч. 1. Параметрическая статистика: учеб. Томск: Изд-во НТЛ, 2012. 540 с.

8. **Кендалл М., Стьюарт А.** Статистические выводы и связи. М.: Наука, 1973.

9. **Айвазян С. А., Енюков И. С., Мешалкин Л. Д.** Прикладная статистика. М.: Финансы и статистика, 1985. 488 с.

Simulation of Society Information Space Evolution Process (The Meme-Gram-Model)

A. A. Artemov, artemsince2@ya.ru, Center of Strategic Nuclear Forces Research at Academy of Military Sciences, Moscow Region, Yubileiny, 141090, Russian Federation, **A. V. Galatenko,** agalat@msu.ru, Mechanics and Mathematics Faculty of Lomonosov Moscow State University, Moscow, 119234, Russian Federation

Corresponding author:

Artemov Artem A., Research Associate, Center of Strategic Nuclear Forces Research at Academy of Military Sciences, Moscow Region, Yubileiny, 141090, Russian Federation,
E-mail: artemsince2@ya.ru

Received on September 04, 2017
Accepted on September 18, 2017

The article presents a model and an algorithm for predicting the number of copies of a meme in the future content of the society's information space (IP). Authors presents a hypothesis that the information space of contemporary society can be described by the dynamics of changing of informational messages, which are spread by the means of mass communication technologies. The prediction period of the model is one calendar week for every three values of number of meme copies in three previous weeks. The model is statistical and was made based on the analysis of changes in the content of Russian-speaking segment of information space from May to August 2014 presented by a collection of time-varying text corpuses which were generated from 300 000 most cited in social media publications of various Internet media. In the article authors evaluate the quality of the model, which show the probability of errors in the forecast of number of copies of a meme. A special feature of the algorithm is the possibility of its application for prediction of number of meme copies for any IP with similar size and diversity parameters to the Russian segment of IP. The results of the study can be summarized in the following statements: 1) based on the analysis of the changes dynamics in the number of 200 000 bigrams copies or memes (training sample), we have developed a mathematical model that solves the problem of predicting the number of meme copies in IP for the next weekly interval depending on the change in the number of copies of the meme in the 3 previous weekly intervals; 2) a theorem on the distribution of the number of meme copies in the next discrete period of time p + 1 has been produced and proved; 3) validation of the constructed statistical model of the number of meme copies distribution in the next discrete moment of time on real data showed a high accuracy of the actual values of the response within the given 95 % confidence intervals with an average error of less than 1 %. Quality assessment of the developed model (algorithm) has been obtained as a result of conducting 100,000 experiments to forecast the number of copies of a meme in the control sample (validation set).

Keywords: *Mem-Gram-Model, language model, n-gram, bigram, memetic, meme, information space, Su-Johnson's distribution, distribution law, number of memes' copies, statistical model, algorithm.*

For citation:

Artemov A. A., Galatenko A. V. Simulation of Society Information Space Evolution Process (The Meme-Gram-Model), *Programmnaya Ingeneria*, 2017, vol. 8, no. 11, pp. 511—523.

DOI: 10.17587/prin.8.511-523

References

1. **Ramisch C.** N-gram models for language detection, available at: http://en.scribd.com/doc/244210124/N-gram-Models-for-Language-Detection#.
2. **Mikolov T., Chen K., Corrado G., Dean J.** Efficient estimation of the world representation in vector space, available at: http://arxiv.org/pdf/1301.3781.pdf
3. **Leskovec J., Backstrom L., Kleinberg J.** Meme-tracking and the dynamics of the news cycle, *Proceedings of the 15th ACM SIGKDD — International Conference on Knowledge Discovery and Data Mining,* New York, ACM, 2009, pp. 497—506.
4. **Gabora L.** Meme and variations: A computational model of cultural evolution, *1993 Lectures in complex systems*, Addison Wesley, 1995, pp. 471—485.
5. **Artemov A. A.** Eksperiment po modelirovaniju processa jevoljucii soderzhanija informacionnogo prostranstva sociuma (s primeneniem mem-gramm-modeli) (An experiment of simulation the process of society information space content evolution (using the meme-gram-model)), *Programmnaya Ingeneria,* 2016, vol. 7, no. 7, pp. 291—306 (in Russian).
6. **Artemov A. A.** Model' ocenki urovnja ugroz informacionnyh vyzovov planu soderzhanija informacionnogo prostranstva social'no-telekommunikacionnoj sistemy (A model for assessing the level of threats to information calls to the content plan of the information space of the socio-telecommunications system), *Informacioniy vojni*, 2015, no. 3, pp. 83—97 (in Russian).
7. **Shulenin V. P.** *Matematicheskaja statistika. CH. 1. Parametricheskaja statistika: uchebnik* (Math statistics. Part 1. Parametric statistics: a textbook), Tomsk, Publishing house of NTL, 2012, 540 p. (in Russian).
8. **Kendall M., Stewart A.** *Statisticheskie vyvody i svjazi* (Statistical conclusions and connections), Moscow, Nauka, 1973 (in Russian).
9. **Ayvazyan S. A., Enyukov I. S., Meshalkin L. D.** *Prikladnaja statistika* (Applied statistics), Moscow, Finance and Statistics, 1985, 488 p. (in Russian).

УДК 51-74 DOI: 10.17587/prin.8.524-528

Д. Ж. Сайфутдинов, студент, e-mail: densssovv@yandex.ru, ОТИ НИЯУ МИФИ, г. Озерск, Челябинская обл.

Геолокационная система наблюдения и анализа передвижения транспортных средств в реальном времени

В настоящее время в открытом доступе существуют различные геоинформационные системы, позволяющие строить маршруты передвижения по городу на общественном транспорте либо предоставляющие сведения о местонахождении общественного транспорта в текущий момент времени. Существенным недостатком таких систем является то, что при построении маршрутов не учитывается текущая ситуация на дорогах, а также не используются геолокационные данные транспорта. В настоящей статье представлена разработанная автором программная система, позволяющая строить маршруты передвижения на общественном транспорте с учетом их геолокационных данных. Демоверсия системы доступна по адресу http://gsmd.000webhostapp.com/www/index.php.

Ключевые слова: *геоинформационная система, геолокационные данные, маршруты передвижения, общественный транспорт, данные реального времени*

Введение

Жители больших городов много времени затрачивают на дорогу и зачастую им приходится совершать пересадки с одного маршрутного транспортного средства на другое. Ожидание такого транспорта, пробки на дорогах, расстояние — все это влияет на время, проведенное в пути.

В сети Интернет существует большой выбор различных сервисов для определения загруженности дорог и расположения транспортных средств. Практически каждый, у кого есть мобильное устройство, может в любой момент времени определить, где находится автобус, трамвай или троллейбус, а также составить маршрут движения по городу.

В качестве примеров можно рассмотреть такие сервисы, как 2ГИС и Яндекс.Транспорт. Задачей этих сервисов является построение маршрутов передвижения по городу с помощью общественного транспорта. Проведя несколько тестов, можно сказать, что 2ГИС и Яндекс.Транспорт строят кратчайший маршрут передвижения, рассчитывая минимальное расстояние проезда. Однако для нахождения наиболее быстрого маршрута передвижения необходимо учитывать не только длину пути, но и местонахождение транспорта по маршруту, а также транспортную ситуацию в городе.

В связи с изложенным выше можно выделить следующие недостатки рассматриваемых систем, которые характерны и для других систем подобного назначения:

- при построении маршрутов не учитывается местоположение маршрутных транспортных средств;
- некоторые сервисы выполняют только одну функцию — либо прокладывают маршрут, либо отображают местоположение маршрутных транспортных средств.

В результате анализа существующих систем автором настоящей статьи было принято решение создать геоинформационную систему, которая обеспечивает пользователей достоверной информацией о местонахождении транспортных средств и формирует оптимальные маршруты для передвижения. По результатам библиографических исследований аналогов информационной системы с такими свойствами найдено не было.

Суммируя сказанное выше, разрабатываемая система должна:

- получать и обрабатывать геолокационные данные транспортного средства;
- отображать местоположение транспортного средства и его характеристики на карте города;
- отображать примерное время прибытия транспортного средства к заданному месту на этой карте;
- строить маршрут передвижения с учетом местоположения транспортного средства и текущей ситуации на дороге.

Программная реализация системы

Доступ к информационной системе осуществляется посредством мобильного приложения либо веб-приложения. Мобильное приложение для ОС Android позволяет пользоваться системой в любой точке города при наличии доступа к сети Интернет. Веб-приложение доступно через браузер. Разработанная система использует карты Google Maps (рис. 1).

Разработанная система состоит из следующих трех частей (рис. 2):

1) клиент (мобильное приложение и веб-страница);

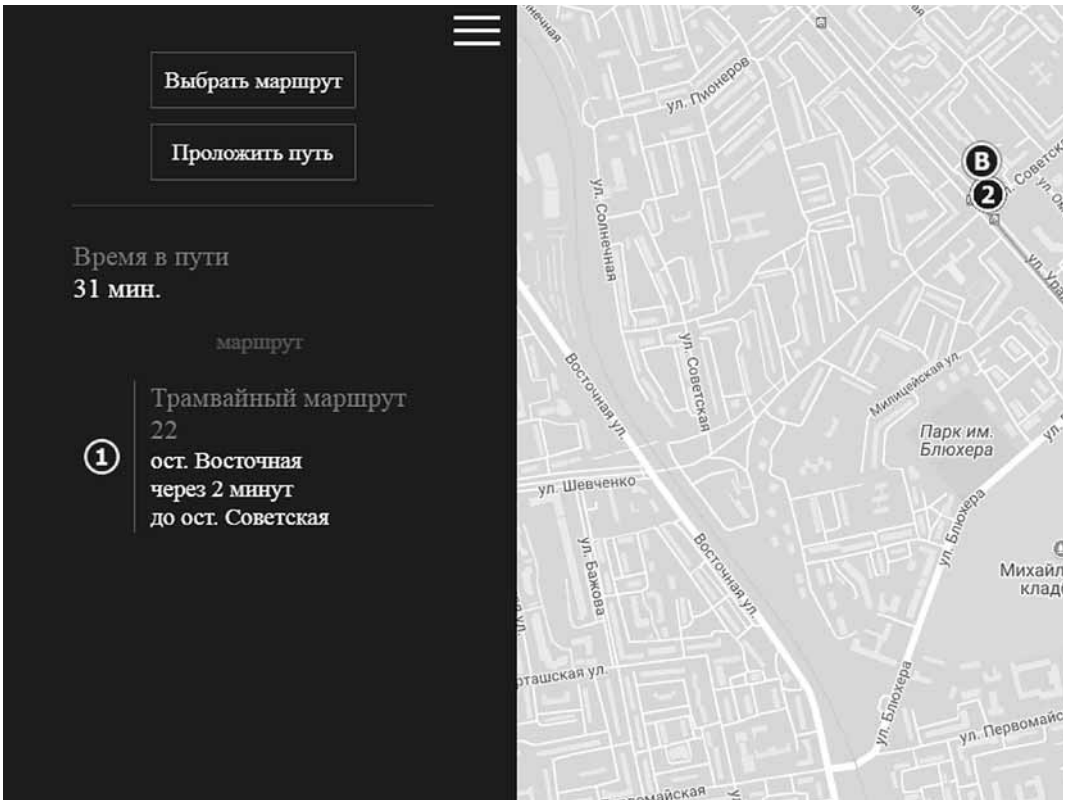

Рис. 1. Интерфейс веб-клиента

2) сервер (реализует составленный алгоритм и хранит базу данных);

3) сторонние серверы (предоставляют данные о транспортных средствах и их маршрутах).

Используемые средства разработки:

- Zend Studio: PHP — для разработки сервера;
- Zend Studio: HTML, CSS, JS — для разработки веб-клиента;
- Android Studio: Java — для разработки Android-приложения.

Разработанная в ходе проектирования база данных используется для хранения информации о маршрутах и остановках транспортных средств. Основными объектами модели данных являются:

- тип транспорта (наименование типа транспорта);
- маршруты (номер маршрута, его название и среднее время полного проезда маршрута);
- точки на маршруте (название точки и ее координаты, порядковый номер точки на маршруте и статус (остановка или нет)); на основании точек на карте строится маршрут; некоторые точки являются остановками.

Для заполнения базы данных используется PHP-скрипт, который получает и записывает/обновляет данные о маршрутных транспортных средствах, их маршрутах и остановках.

Составленный и реализованный на сервере алгоритм поиска оптимального построения пути основан на поиске цепочек маршрутов и остановок для пересадки [1]. Выделим следующие основные этапы алгоритма:

1) определение маршрутов в начальной и конечной точке;

2) формирование различных цепочек маршрутов;

3) определение оптимальных остановок для пересадки;

4) наложение данных реального времени на полученные цепочки.

Рис. 2. Архитектура системы

Назовем начало пути точкой *A*, конец пути — точкой *B*.

Этап 1: по заданным координатам точек *A* и *B* определение ближайших, в радиусе 500...700 м остановок, после чего определение номеров маршрутов, которые проходят через эти остановки.

Этап 2: разделение полученных номеров маршрутов на две группы (рис. 3): номера маршрутов начала пути и конца пути.

Далее алгоритм находит возможные варианты маршрутов следующим образом.

Для каждого пути строится дерево, вершины которого — номера маршрутов, а ребра — пересадки. Построение деревьев осуществляется в ширину.

В первую очередь рассматриваются номера маршрутов начала пути — x_1, x_2, ..., x_l. Каждый номер маршрута начала пути сравнивается с номерами маршрутов конца пути, если появляются совпадения, то дальнейшее рассмотрение узла прекращается.

Если маршруты x_2,..., x_l позволяют добраться до конечной точки без пересадок, то поиск дочерних узлов дерева прекращается.

Если по маршруту x_1 невозможно добраться до конечной точки без пересадок, то определяются номера маршрутов, на которые можно совершить пересадку. После этого найденные номера записываются в дочерние узлы дерева (на рис. 3 это маршруты n_1 и n_2). Все найденные варианты маршрутов запоминаются для дальнейшей обработки.

Если требуется две и более пересадок, алгоритм приступает к рассмотрению следующих дочерних узлов и повторяет описанные действия.

Работа алгоритма поиска возможных цепочек маршрутов путей от точки *A* до точки *B* завершается при нахождении комбинаций с минимальным числом пересадок. В данном случае следующие дочерние узлы рассматривают последними, а последующие не анализируются. Это условие было принято с целью экономии расходов на проезд пользователей приложения.

Еще одним условием завершения работы алгоритма является нахождение всех возможных вариантов путей передвижения из точки *A* в точку *B*.

Этап 3: определение для полученных цепочек маршрутов оптимальных остановок для пересадок методом ветвей и границ. Для каждой пересадки определение набора возможных реальных остановок в городе. Сначала выбор для каждой пересадки остановки и подсчет полного расстояния от начала до конца, затем запись ее значения в переменную *min*. Все последующие варианты рассчитываются по частям:

1) выбор начального маршрута и вычисление расстояния до остановки, на которой необходимо совершить пересадку;

2) сравнение полученного расстояния со значением *min*; если полученное расстояние не превышает значения *min*, выбор следующего маршрута в пути и продолжение разбора; как только сумма расстояний рассмотренных отрезков превышает значение *min*, прекращение разбора текущих вариантов остановок и переход к следующему набору остановок;

3) на завершающей стадии разбора пути, если расстояние всех отрезков не превышает значения *min*, то запись значения этого пути в *min* и запоминание самого пути.

Определение всех наборов остановок прекращается после просмотра всех вариантов остановок.

Этап 4: запрос данных о маршрутных транспортных средствах и состоянии транспортного потока в городе.

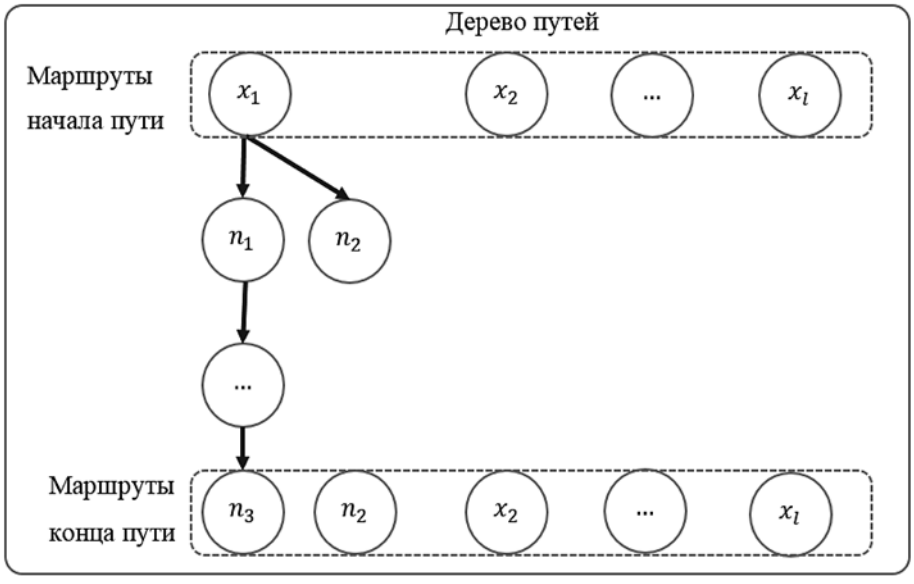

Рис. 3. Определение цепочек маршрутов:

x_1, x_2,..., x_l — номера маршрутов начала пути; n_1, n_2, ..., n_j — номера маршрутов, на которые можно пересесть

Полученные данные со сторонних серверов присылают в формате json, они содержат следующую информацию о транспорте: точная дата и время; азимут транспортного средства; скорость; координаты местоположения и его бортовой номер.

На основе полученных данных определяется направление движения маршрутного транспортного средства по маршруту и его следующая остановка. Для этого необходимо выполнить следующие шаги:

1) по азимуту определить четверть, в область которой направляется транспортное средство;

2) поиск остановок, к которым, предположительно, прибывает транспорт; если таких остановок не существует, то увеличение угла поиска на соседние четверти (рис. 4).

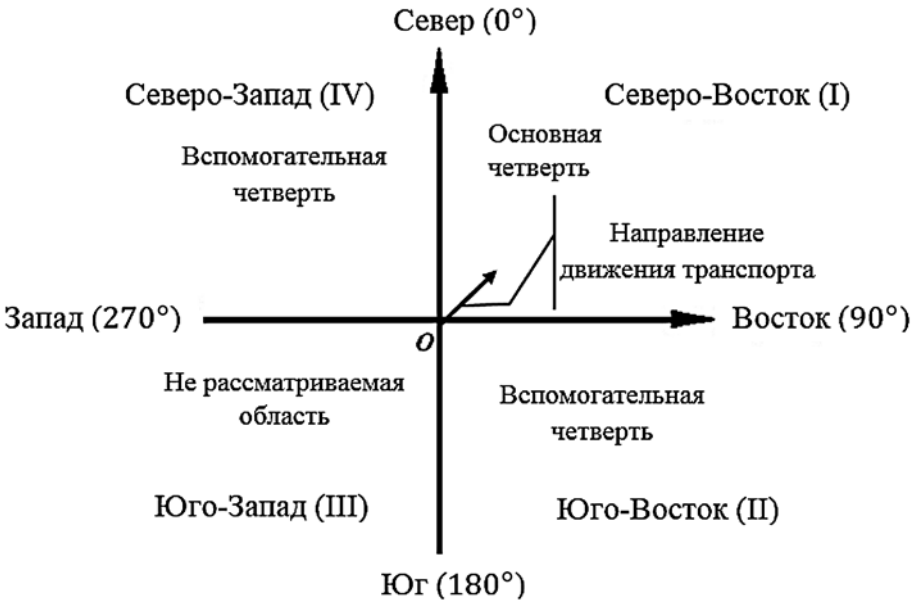

Рис. 4. Область поиска остановок

3) дальнейший анализ ситуации в зависимости от числа найденных остановок:

- если найдена одна остановка, то определение по этой остановке направления движения транспорта по маршруту;
- если найдено две остановки, то определение по координатам этих остановок четверти направления движения; если она совпадает с четвертью направления движения транспортного средства, то определение по остановке направления маршрута; при несовпадении четвертей направление движения транспорта по маршруту считается обратным.
- если найдено три остановки, то выбор двух остановок на одном маршруте и выполнение с ними действий для двух остановок;
- если найдено четыре остановки, то выполнение действий для трех остановок, за исключением одного случая; при равном числе остановок по направлениям для каждого направления выполнение действий для двух остановок.

4) после определения направления движения всех транспортных средств расчет времени их движения до начальной остановки и их дальнейшего времени движения в пути;

5) при расчете времени движения транспорта по маршруту проводятся следующие вычисления: для каждого маршрута определяется среднее время движения транспортного средства на отрезке пути; после этого изменяется скорость транспорта на отрезке — в зависимости от полученной средней скорости.

В результате работы алгоритма происходит формирование до трех возможных вариантов путей передвижения из точки *A* в точку *B*, которые предоставляются пользователю.

Результаты испытания системы

В ходе испытания разработанной системы было проведено функциональное тестирование (тестирование черного ящика). Тестирование черного ящика подразумевает под собой проверку реализации функций системы, без учета внутренней структуры ее кода и без доступа к базе данных. Результаты выполнения функций при определенном наборе данных, подаваемых на вход системы, известны.

Для этого были запущены следующие тесты: на построение маршрутов передвижения в транспортном средстве без пересадок; на построение маршрутов с пересадками; на отображение маршрутов и местонахождения транспортных средств.

Тестирование проводили на маршрутных транспортных средствах Екатеринбурга. В результате выполнения тестов система представила оптимальные маршруты передвижения по городу. Средняя скорость отклика на запрос составила в среднем примерно две секунды.

Заключение

В статье описана разработанная геоинформационная система реального времени.

Разработанная система оперативно прокладывает маршрут для передвижения по городу с помощью общественных транспортных средств в реальном времени, предоставляет информацию о местоположении транспортных средств.

В ходе дальнейшей работы по совершенствованию системы предполагается внести ряд перечисленных далее улучшений:

- добавить прогнозирование состояния транспортного потока в городе;
- добавить возможность работы с временно измененными маршрутами;
- добавить новые транспортные средства;
- реализовать возможности использования в различных городах;
- добавить возможность работы с временно измененными маршрутами.

Список литературы

1. **Сайфутдинов Д. Ж.** Геолокационная система наблюдения и анализа передвижения транспортных средств в реальном времени // XVII всероссийская научно-практическая конференция "Дни науки — 2017". 20—22 апреля 2017. С. 139, URL: http://oti.ru/wp-content/uploads/2015/09/Maket-sbornika-2017.pdf (Дата обращения: 12.07.2017).

Geolocation System for Monitoring and Analyzing the Movement of Vehicles in Real Time

D. Zh. Saifutdinov, densssovv@yandex.ru, OTI NNIU MEPhI, Chelyabinsk region, Ozersk, 456780, Russian Federation,

Corresponding author:

Saifutdinov Denis Zh., Student, OTI NNIU MEPhI, Chelyabinsk region, Ozersk, 456780, Russian Federation, E-mail: densssovv@yandex.ru

Received on September 02, 2017
Accepted on September 11, 2017

Currently, there are, in the open access, various geoinformation systems that allow one to build routes for moving with in a city by public transport, as well as provides information about the location of public transport at the current time. A significant drawback of such systems is that the building of routes does not take into account the current situation on the roads, and also does not use the geolocation data of transport. This article presents a developed geolocation system that allows to build city movement routes taking into account geolocation data. The demo version of the system is available at http://gsmd.000webhostapp.com/www/index.php.

Keywords: *geoinformation systems, geolocation data, public transport, public transport routes, real time data*

For citation:

Saifutdinov D. Zh. Geolocation System for Monitoring and Analyzing the Movement of Vehicles in Real Time, *Programmnaya Ingeneria*, 2017, vol. 8, no. 11, p. 524—528.

DOI: 10.17587/prin.8.524-528

Reference

1. **Saifutdinov D. Zh.** Geolokacionnaja sistema nabljudenija i analiza peredvizhenija transportnyh sredstv v real'nom vremeni (Geolocation system for monitoring and analyzing the movement of vehicles in real time), *XVII Russian scientific and practical conference "Dni nauki — 2017"*. April 20—22, 2017, p. 139, available at: http://oti.ru/wp-content/uploads/2015/09/Maket-sbornika-2017.pdf (in Russian).

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *Ю. Ф. Кравчинская*

Сдано в набор 12.09.2017 г. Подписано в печать 23.10.2017 г. Формат 60×88 1/8. Заказ PI1117
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**