DOI 10.37882/2223–2966.2021.05–2.15

АВТОМАТИЗИРОВАННЫЙ ПОДХОД К ОБНАРУЖЕНИЮ СЕМАНТИЧЕСКИ БЛИЗКИХ ЗАПРОСОВ ЗАКАЗЧИКА В СИСТЕМЕ ОТСЛЕЖИВАНИЯ ОШИБОК JIRA

AN AUTOMATED APPROACH TO FINDING SEMANTICALLY RELATED CUSTOMER REQUESTS IN THE JIRA BUG TRACKING SYSTEM

A. Kovalev
I. Nikiforov
P. Drobintsev

Summary. The work is devoted to research in the field of software maintenance phase automation. An automated approach to solving customer requests is proposed, which consists in using the Doc2Vec algorithm to search for semantically similar solved requests, as well as to find competent software engineers in the Jira issue tracking system. The proposed approach is implemented in a software tool, which allows to reduce the labor intensity of the maintenance stage by 12%. The results compare the manual and automated approach to analyzing customer requests in the process of software product support.

Keywords: software maintenance, automation, Doc2Vec, machine learning.

Ковалев Артем Дмитриевич
Аспирант, ассистент, Санкт-Петербургский политехнический университет Петра Великого
kov3000@ya.ru
Никифоров Игорь Валерьевич
К.т.н., доцент, Санкт-Петербургский политехнический университет Петра Великого
i.nikiforov@ics2.ecd.spbstu.ru
Дробинцев Павел Дмитриевич
К.т.н., доцент, Санкт-Петербургский политехнический университет Петра Великого
drob@ics2.ecd.spbstu.ru

Аннотация. Работа посвящена исследованию в области автоматизации этапа сопровождения программного обеспечения. Предложен автоматизированный подход решения запросов заказчика, который заключается в использовании алгоритма Doc2Vec для поиска семантически близких решенных запросов, а также для нахождения компетентных инженеров-разработчиков в системе отслеживания ошибок Jira. Предложенный подход реализован в программном средстве, которое позволяет снизить трудоемкость этапа сопровождения на 12%. В результатах приведено сравнение ручного и автоматизированного подхода к анализу запросов заказчика в процессе сопровождения программного продукта.

Ключевые слова: сопровождение, автоматизация, Doc2Vec, машинное обучение.

Введение

Этап сопровождения программного обеспечения (ПО) является трудоемким и занимает по оценкам [1] более 70% всех затрат жизненного цикла ПО. На данном этапе инженер технической поддержки принимает и обрабатывает запросы от заказчиков программного продукта. Запросы могут приходить по электронной почте, с помощью телефонной связи или через HelpDesk системы [2]. Запрос, как правило, написан на естественном языке и содержит описание проблемы в программном продукте, а также прикрепленные файлы: конфигурации, логи, трассировки стека, скриншоты, видеозаписи воспроизведения проблемы. Когда инженеры технической поддержки не могут обработать запрос заказчика, то они передают его инженеру-разработчику. Для хранения и управления такими запросами используются системы отслеживания ошибок [3]. В данной работе рассмотрена система Jira, которая является одним из самых популярных инструментов в своей области.

При решении запросов заказчика часто необходимо найти похожие уже решенные запросы, а также привлечь компетентных инженеров-разработчиков, которые специализируются в соответствующей проблемной области. В решении данных задач могут помочь алгоритмы машинного обучения, например, Doc2Vec [4]. За счет использования программного средства, основанного на алгоритме Doc2Vec, возможно снижение трудоемкости и повышение эффективности процесса сопровождения. В результате повышение качества сопровождения помогает выстроить долгосрочные отношения поставщика ПО с заказчиком.

Таблица 1. Сравнительный анализ инструментов

Критерий/Инструмент	Toolkit	Automation	ScriptRunner
Анализ комментариев	–	+	+
Анализ похожих заявок	–	–	–
Предоставление рекомендаций	+	–	–
Редактирование заявки	–	+	+

Обзор литературы

Одно из первых исследований по обнаружению семантически похожих отчетов об ошибках принадлежит L. Hiew [5]. Предлагаемый подход основан на преобразовании текста в слово-вектор с использованием операции стемминга и техники TF-IDF [6]. Для измерения величины, которая характеризует схожесть двух отчетов об ошибках, к сформированным векторам документов было применено косинусное сходство.

C. Sun и др. [7] предложили дискриминантную модель с использованием машины опорных векторов (SVM) для обучения модели на основе набора помеченных векторов. Затем эта модель была использована для обнаружения дубликатов отчетов об ошибках.

B. Nguyen и др. [8] в своем исследовании предложили подход, который использует преимущества как функций на основе подхода IR (BM25F), так и функций на основе подхода разбиения на темы (LDA).

Ни в одном из рассмотренных исследований не используется подход, основанный на алгоритме Doc2Vec. Таким образом, отличительной особенностью нашего исследования является использование данного алгоритма для выявления семантически похожих запросов клиентов.

Существующие решения

Основным инструментом для работы с заявками в системе отслеживания ошибок Jira являются плагины. На текущий момент их количество составляет около 1500. В данной работе рассматриваются наиболее популярные плагины, использующиеся для автоматизации процесса сопровождения и получения дополнительной информации по заявкам.

Toolkit Plugin — плагин, разработанный для определения нерешенных запросов, в которых не было активности в течение 7 дней, а также для добавления информационных колонок с данными о количестве комментариев и приложенных документов.

ScriptRunner — плагин, который позволяет создавать скрипты для событий в системе Jira. Например, создание заявки или перевод заявки в другой статус. Плагин работает с Jira как с внешней системой путем отправки запросов на API.

Automation — плагин, реагирующий на события в Jira. Например, изменение одного из полей заявки или добавление нового комментария. В отличии от ScriptRunner не требует написания скриптов, так как имеет графический интерфейс для настройки правил.

Для сравнительного анализа существующих средств, используются следующие критерии:

- *Анализ комментариев.* Рассматривается возможность семантического анализа комментариев, нахождения ключевых слов или ключевых фраз в комментарии.
- *Анализ похожих заявок.* Поиск семантически близких заявок.
- *Предоставление рекомендаций.* Предложения по работе с заявкой на основе выполненного анализа. Например, предложение перевести заявку на другого инженера.
- *Редактирование заявки.* Применение рекомендаций или заранее прописанных в правилах изменений заявки в автоматическом режиме. Например, при заведении новой заявки автоматически переводить ее на менеджера отдела разработки.

Для каждого из критериев введем следующие характеристики:

- не поддерживается: –
- полностью поддерживается: +

В таблице 1 приведено сравнение существующих решений.

Из таблицы 1, можно сделать вывод что для автоматизации обработки запросов заказчиков больше всего подходят плагины Automation и ScriptRunner. Но данные программные средства не предоставляют анализ семантической схожести заявок заказчика, поэтому

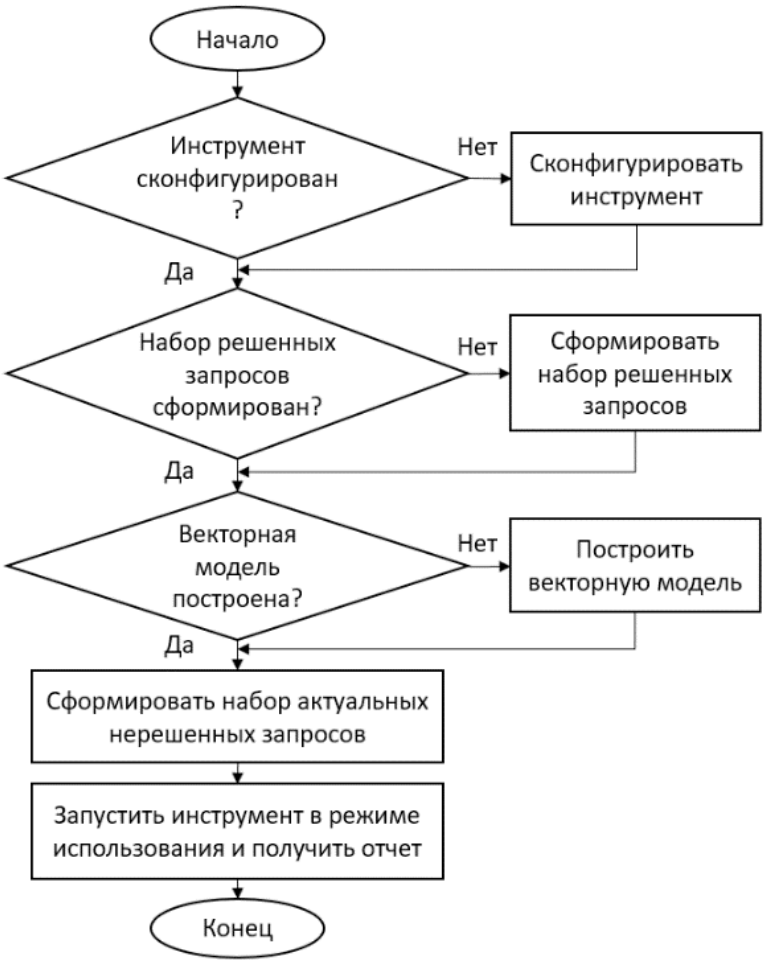

Рис. 1. Блок-схема автоматизированного подхода

необходимо разработать автоматизированный подход, отличительной особенностью которого является применение алгоритма для поиска семантически близких запросов.

Предлагаемый подход

В данной работе предлагается снизить трудоемкость ручного анализа запросов заказчика. Для этого необходимо описать сценарий ручного подхода, выделить те моменты, которые могут быть автоматизированы и предложить свой автоматизированный подход. Предлагаемый подход должен быть реализован в программном средстве.

Ручная обработка запросов

Суть ручного подхода состоит в том, чтобы получить список нерешенных запросов заказчика в системе отслеживания ошибок, а затем итеративно рассматривать и обрабатывать каждый запрос.

Сценарий обработки одного нерешенного запроса представлен ниже:

1. Прочитать заголовок, описание, шаги воспроизведения проблемы и комментарии.
2. Понять смысл запроса и оценить сложность.
3. Посмотреть количество прикрепленных файлов, их названия и содержимое.
4. На основе комментариев оценить актуальный статус запроса. В случае, если он не соответствует текущему статусу, то поменять его.
5. Если инженеру-разработчику необходима дополнительная информация, то запросить ее у инженера технической поддержки в комментариях.
6. Установить имя инженера-разработчика, ответственного за область, в которой произошла ошибка.
7. Попытаться определить похожий решенный запрос в системе отслеживания ошибок или в базе знаний.
8. Если решение запроса известно, то предоставить необходимую информацию инженеру технической поддержки.

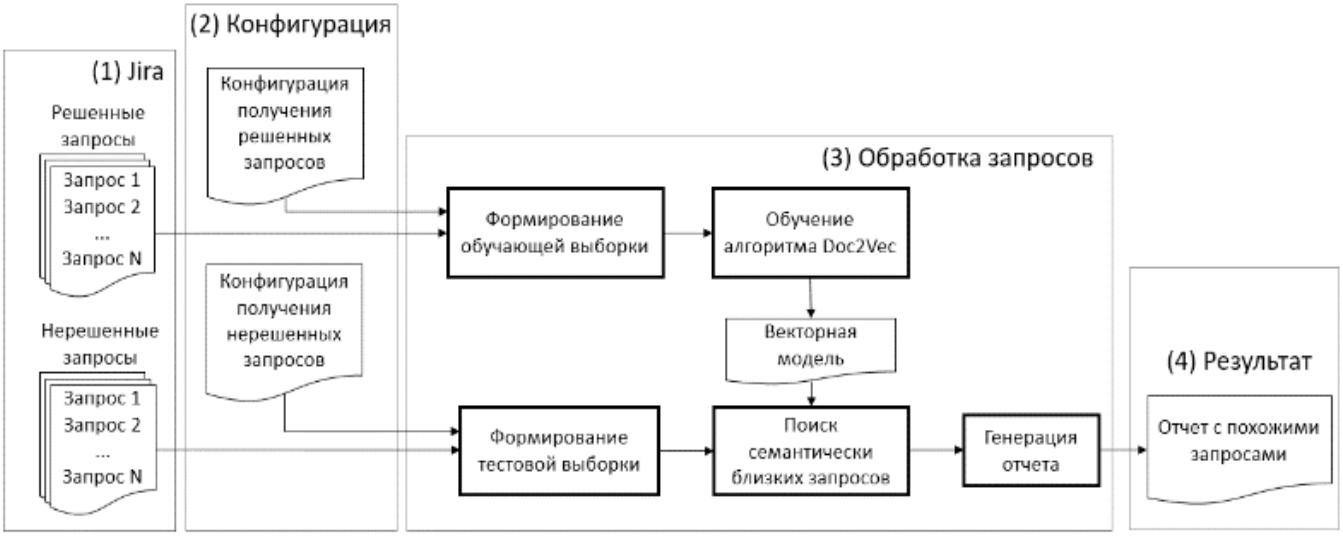

Рис. 2. Концептуальная схема предлагаемого подхода к обработке запросов заказчика

Каждый инженер-разработчик выполняет данную последовательность действий при обработке тех запросов, которые были ему поручены. Для снижения трудоемкости процесса сопровождения, некоторые пункты из представленного сценария можно автоматизировать.

Автоматизация обработки запросов

В работе предлагается автоматизировать часть ручного процесса анализа заявок заказчика, а именно 6-й и 7-й пункт из списка выше. Для этого необходимо разработать программный инструмент, который сможет решить задачу автоматизации. Сценарий работы над запросами с использованием разработанного подхода представлен в виде блок-схемы на рис. 1.

На блок-схеме алгоритма видно, что если программный инструмент сконфигурирован, набор решенных вопросов сформирован, а векторная модель построена, то алгоритм обработки запросов заказчика сокращается до двух действий: формирование набора нерешенных запросов и запуска инструмента для анализа запросов.

Концептуальная схема

Рассмотрим предлагаемую концептуальную схему анализа запросов заказчика, изображенную на рис. 2.

Схема разделена на блоки и состоит из системы отслеживания ошибок Jira, в которой хранятся запросы заказчика (1), конфигурационных файлов для формирования наборов решенных и нерешенных запросов (2), разрабатываемого программного средства для анализа нерешенных запросов и поиска семантически близких решенных запросов (3), а также итогового отчета, в котором для каждого нерешенного запроса подобрано несколько уже решенных, а также указан компетентный инженер-разработчик (4).

Система отслеживания ошибок Jira предоставляет REST API интерфейс для получения данных, таких как, описания, шаги воспроизведения, комментарии и другие поля запроса. С помощью параметров поискового запроса через REST интерфейс Jira можно получить необходимые для обработки запросы заказчика.

Конфигурации получения решенных и нерешенных запросов состоят из параметров, с помощью которых происходит подключение к интерфейсу Jira. Например, адрес ресурса, таймаут соединения, количество получаемых запросов заказчика за одно обращение к серверу и т.д.

Сформированный набор решенных запросов используется алгоритмом машинного обучения Doc2Vec для создания векторной модели. После процесса обучения модель, хранящаяся в оперативной памяти, сохраняется на диск в виде файла. Эту модель необходимо создать на этапе настройки системы, чтобы в дальнейшем ее использовать для нахождения семантически близких запросов.

Для поиска семантически близких запросов заказчика необходимо с помощью алгоритма Doc2Vec произвести векторизацию набора нерешенных запросов. В результате из каждого нерешенного запроса формируется числовой вектор. После этого определяется схожесть векторов каждого нерешенного запроса с векторами уже решенных запросов. В данном исследовании в качестве меры схожести двух векторов предлагается исполь-

```
AMQ-6974 | overflow errors with timestamps Some comparisons with timestamp values are not safe This comparisons can trigger errors
AMQ-6973 | ActiveMq get stuck after 2-3 dayz After every 2 3 days we need to bounce activemq services with below error message INFO
AMQ-6972 | Original destination is not accessible via AMQP when using Virtual topics ActiveMQ MQTT transport is configured for virt
AMQ-6971 | Active MQ server connection.start() activemq xml udp transportConnector name udp uri udp 0 0 0 0 9010 java code import o
AMQ-6970 | SSL config-params are not propagated inside rar correctly When trying to configure the SSL config properties such as key
AMQ-6969 | Number Of Pending Messages Did Not Reflect Its Queues Number Having issue on number of pending messages in any queues cr
AMQ-6967 | Periodic expiry with no consumers fails to page in messages if cache has flipped when there are no consumers on a queue
AMQ-6966 | mqtt over wss impossible get certificate details steps connect from paho mqtt client to mqtt ssl transport in activemq f
AMQ-6965 | Failed to fill batch | org.apache.activemq.broker.region.cursors.AbstractStoreCursor | Scheduler java.io.EOFException: C
AMQ-6964 | Store COMMIT FAILED: ClassCastException C12018 05 09 02 35 17 497 WARN Store COMMIT FAILED org apache activemq transacti
AMQ-6963 | ActiveMQConnection: fix logging for avoid volatile read in case of different loglevel Just a little improvement for Acti
AMQ-6961 | ActiveMQ broker does not detect dead clients I have a system formed by 1 client producer on an AMQ queue which uses Open
AMQ-6954 | Queue page on web console displays URL parameter without proper encoding Using a URL with the parameter QueryFilter the
AMQ-6951 | Hide embedded jetty version Hi sorry in advance if this is something easy for jetty experts We need some guidance or see
AMQ-6949 | SocketTimeoutException when using HTTP transport connector Java clients that connect to the ActiveMQ broker over the HTT
```

Рис. 3. Часть набора решенных запросов в файле ResolvedDataSet.txt

зовать косинусное сходство [9]. Из решенных запросов, представленных наиболее схожими векторными представлениями, извлекаются идентификаторы, а также имена инженеров-разработчиков, которые решили проблему.

В результате применения алгоритма Doc2Vec к нерешенным запросам генерируется отчет, который представляет из себя текстовый файл. Каждая строчка этого файла соответствует нерешенному запросу заказчика. В начале строки идет идентификатор нерешенного запроса, а потом через запятую идентификаторы похожих решенных запросов и имена компетентных инженеров-разработчиков.

Реализация программного инструмента

Разработанный инструмент, который реализует предлагаемый автоматизированный подход, написан на языке Java 8 и состоит из двух модулей: Jira-коннектора и модуля поиска семантически близких запросов.

Для загрузки данных из Jira и формирования файлов с наборами запросов был разработан и использован Jira-коннектор. Подключение к Jira осуществляется с помощью REST API интерфейса. В зависимости от входных параметров можно получить как решенные, так и нерешенные запросы из любого проекта. Ответы от сервера Jira в формате JSON преобразуются и записываются в файлы ResolvedDataSet.txt и UnresolvedDataSet.txt. При этом каждая строка файла соответствует одному запросу. Формат создаваемых файлов, содержащих решенные и нерешенные запросы, можно увидеть на рис. 3.

В начале каждой строки стоит идентификатор запроса. После него стоит разделяющий символ вертикальной черты. За этим символом следуют текстовые данные, определяющие семантику запроса. В качестве текстовых данных запроса используются поле заголовка (summary), описания (description) и шагов воспроизведения (steps to reproduce).

Основу программного средства составляет алгоритм Doc2Vec, который реализован в Java библиотеке Deeplearning4j [https://deeplearning4j.org].

Для обучения алгоритма Doc2Vec используется файл ResolvedDataSet.txt сформированный Jira коннектором и содержащий предварительно обработанные тексты запросов. В результате обучения получается векторная модель, которая в последующем сериализуется в файл VectorModel.zip. Вектора представляют смысл запросов и с помощью математических операций над векторами можно обнаружить сходство между разными запросами.

После обучения алгоритма Doc2Vec можно приступить к его использованию. Для этого в память подгружается модель из файла VectorModel.zip, в которой каждый запрос представлен в виде числового вектора и сопоставлен с идентификатором запроса. После этого на вход алгоритма Doc2Vec подаются нерешенные запросы из файла UnresolvedDataSet.txt.

Таким образом, результатом работы алгоритма является список ключей решенных запросов наиболее похожих на входные нерешенные запросы. Далее из найденных похожих запросов происходит извлечение дополнительных полей, например имена инженеров-разработчиков, вовлеченных в решение запроса. Все эти данные затем формируются в отчет и предоставляются пользователю.

Эксперимент

Разработанный инструмент применен на проекте Apache ActiveMQ [https://issues.apache.org/jira/projects/AMQ]. Проект является свободно распространяемым и любой пользователь сети Интернет может зарегистрировать запрос, содержащий вопрос или ошибку,

в системе отслеживания ошибок Jira. Для обучения и тестирования модели Doc2Vec использован набор Jira-запросов, взятый за период времени с 20 апреля 2004 года по 15 ноября 2020 года, что в итоге составляет 7100 запросов.

Весь набор данных разделен на две части: обучающую выборку (90%, или 6390 запросов), представляющую решенные запросы и тестовую выборку (10%, или 710 запросов), представляющую нерешенные запросы. Обе этих выборки располагаются в двух соответствующих файлах: ResolvedDataSet.txt и UnresolvedDataSet.txt.

В процессе работы над проектом Apache ActiveMQ проведен один эксперимент. Суть эксперимента заключается в сравнении среднего времени анализа одного запроса заказчика автоматизированным и ручным подходом. В данном случае запрос является проанализированным, если найден похожий по смыслу решенный запрос и компетентный инженер-разработчик.

Сначала была произведена настройка программного средства. На настройку конфигурационных файлов потребовалось 5 минут. На этапе подготовки инструмента выгружено 7100 запросов. Время выгрузки заняло 2 часа. Файл ResolvedDataSet.txt с решенными запросами, которые являются обучающей выборкой для алгоритма Doc2Vec, составляет 514 Мб. Файл UnresolvedDataSet.txt с нерешенными запросами, которые являются тестовой выборкой, составляет 49 Мб. Затем сформирована векторная модель, которая сохранена на диск в файле VectorModel.zip. Размер файла модели составляет 287 Мб. Создание модели заняло 36 минут.

В рамках эксперимента проанализировано 30 запросов из тестовой выборки. Для анализа 30 запросов проекта Apache ActiveMQ ручным способом потребовалось 368 минут, а для анализа автоматизированным подходом потребовалось 324 минуты. При этом выигрыш по времени составил 12%.

При ручном подходе время, потраченное на анализ одного запроса, зависит от его сложности, количества полезной информации в запросе и опыта разработчика, который занят решением данной проблемы. В автоматизированном подходе время настройки и качество анализа напрямую зависят от количества уже решенных запросов в проекте.

Заключение

В ходе работы проведен обзор существующих подходов для автоматизации сопровождения программных продуктов в системе отслеживания ошибок Jira. Обоснована цель создания автоматизированного программного инструмента для анализа запросов заказчика.

Предложен автоматизированный подход, позволяющий сократить трудоемкость рассмотрения запросов заказчика. Этот подход основан на использовании алгоритма машинного обучения Doc2Vec, который решает задачу поиска семантически близких запросов.

Созданный инструмент успешно протестирован на проекте Apache ActiveMQ. В результате применения инструмента проанализировано 30 запросов. Показана эффективность его использования.

Представлены преимущества использования программного средства. Время анализа одного запроса снизилось по сравнению с традиционным ручным подходом. Это произошло за счет уменьшения трудоемкости поиска похожих запросов и компетентных инженеров на 12%.

Анализ запросов проводимый с помощью разработанного программного средства является более точным и эффективным, чем рассмотрение запросов ручным подходом. Инженер может не помнить или не знать некоторых решенных ранее запросов. В то же время в процессе работы инструмента происходит обращение ко всем историческим данным. В связи с этим повышается вероятность найти похожий решенный ранее запрос.

Скорость и точность рассмотрения запроса напрямую влияют на качество обслуживания заказчика, а, следовательно, и на долгосрочные отношения с ним. Использование программного инструмента, созданного для анализа запросов заказчика, позволяет заменить ручной подход на автоматизированный. При этом повышается качество процесса сопровождения для заказчика и снижается трудоемкость данного процесса для разработчика.

ЛИТЕРАТУРА

1. Ogheneovo E.E. On the relationship between software complexity and maintenance costs // Journal of Computer and Communications, vol. 2, no. 14, p. 1, 2014.
2. Котовсков Я.В. Автоматизация службы тех.поддержки в IT-компании // Актуальные проблемы современной науки. Челябинск: Изд-во Научно-исследовательский центр «Антровита», 2017. С. 102–106.
3. Bertram D., Voida A., Greenberg S., and Walker R. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams // Proceedings of the 2010 ACM conference on Computer supported cooperative work, pp. 291–300, February 2010.

4. Maslova N., Potapov V. Neural network Doc2Vec in automated sentiment analysis for short informal texts // Lecture Notes in Computer Science, 2017. Vol. 10458. pp. 546–554.
5. Hiew L. Assisted detection of duplicate bug reports. University of British Columbia, 2006.
6. Яцко В.А. Достоинства и недостатки взвешивания терминов по формуле TF-IDF // В мире научных открытий. Красноярск: Изд-во ООО «Научно-инновационный центр», 2013. № 6 (42). С. 229–244.
7. Sun C., Lo D., Wang X., Jiang J., and Khoo S.-C. A discriminative model approach for accurate duplicate bug report retrieval // Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, 2010, pp. 45–54.
8. Nguyen A.T., Nguyen T.T., Nguyen T.N., Lo D., and Sun C. Duplicate bug report detection with a combination of information retrieval and topic modeling // Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, 2012, pp. 70–79.
9. Giller G.L. The Statistical Properties of Random Bitstreams and the Sampling Distribution of Cosine Similarity // Giller Investments Research Notes, 2012. No. 20121024/1.

© Ковалев Артем Дмитриевич (kov3000@ya.ru),
Никифоров Игорь Валерьевич (i.nikiforov@ics2.ecd.spbstu.ru), Дробинцев Павел Дмитриевич (drob@ics2.ecd.spbstu.ru).
Журнал «Современная наука: актуальные проблемы теории и практики»

Санкт-Петербургский политехнический университет Петра Великого