Автоматизация и моделирование в проектировании и управлении. 2026. № 2 (32). С. 27-33.
ISSN 2658-3488 print, ISSN 2658-6436 online
Automation and modeling in design and management. 2026. № 2 (32). P. 27-33.

Математическое и компьютерное моделирование

Научная статья
Статья в открытом доступе
УДК 004.89
doi: 10.30987/2658-6436-2026-2-27-33

РАЗРАБОТКА МЕТОДОЛОГИИ КЛАССИФИКАЦИИ КОММИТОВ В *GIT*-РЕПОЗИТОРИЯХ С ИСПОЛЬЗОВАНИЕМ МАШИННОГО ОБУЧЕНИЯ

Дмитрий Игоревич Копелиович[1], Михаил Андреевич Кургуз[2]✉
[1, 2, 3] Брянский государственный технический университет, г. Брянск, Россия
[1] dkopeliovich@yandex.ru, https://orcid.org/0000-0003-4095-7029
[2] mihail.kurguz2016@yandex.ru, https://orcid.org/0009-0008-1270-8918

Аннотация. *Представлена разработка методологии и программного обеспечения для автоматической классификации коммитов в Git-репозиториях с использованием методов машинного обучения. Предложенный подход сочетает в себе текстовую векторизацию на основе TF-IDF и модель Multinominal Naive Bayes для классификации коммитов по категориям. Подход включает в себя систему активного обучения, которая дает пользователю возможность корректировать предлагаемые классификации, что способствует непрерывному совершенствованию модели. Методология включает предварительную обработку описаний коммитов, извлечение семантических признаков и построение адаптивной классификационной модели. Результаты работы могут быть использованы для повышения прозрачности процессов разработки, анализа историй изменений, анализа и оптимизации кода и автоматизации процессов тестирования и доставки новых модулей разрабатываемого проекта заинтересованным сторонам (CI/CD).*

Ключевые слова: машинное обучение, *Git*, классификации коммитов, активное обучение, *TF-IDF*, *Multinominal Naive Bayes*

Для цитирования: Копелиович Д.И., Кургуз М.А. Разработка методологии классификации коммитов в Git-репозиториях с использованием машинного обучения // Автоматизация и моделирование в проектировании и управлении. 2026. №2 (32). С. 27-33. doi: 10.30987/2658-6436-2026-2-27-33.

Original article
Open Access Article

DEVELOPING A METHODOLOGY FOR CLASSIFYING COMMITS IN *GIT*-REPOSITORIES USING MACHINE LEARNING

Dmitry I. Kopeliovich[1], Mikhail A. Kurguz[2]✉
[1] dkopeliovich@yandex.ru, https://orcid.org/0000-0003-4095-7029
[2] mihail.kurguz2016@yandex.ru, https://orcid.org/0009-0008-1270-8918

Abstract. *This paper presents the development of a methodology and software for automatically classifying commits in Git-repositories using machine learning methods. The proposed approach combines text vectorization based on TF-IDF and the Multinomial Naive Bayes model for classifying commits into categories. The approach includes an active learning system that allows the user to adjust the proposed classifications, facilitating continuous model improvement. The methodology includes preprocessing commit descriptions, extracting semantic features, and building an adaptive classification model. The results of this work can be used to improve the transparency of development processes, to analyze change histories, to analyze and optimize code, and to automate testing and delivery of new modules of the project being developed to stakeholders (Continuous Integration / Continuous Delivery).*

Keywords: machine learning, *Git*, commit classification, active learning, *TF-IDF, Multinomial Naive Bayes*

For citation: Kopeliovich D.I., Kurguz M.A. Developing a Methodology for Classifying Commits in *Git*-Repositories Using Machine Learning. Automation and modeling in design and management, 2026, no. 2 (32). pp. 27-33. doi: 10.30987/2658-6436-2026-2-27-33.

Введение

В современной разработке программного обеспечения (ПО) использование систем контроля версий, таких как *Git*, стало стандартом [1]. Репозитории *Git* содержат историю изменений кода в виде коммитов, каждый из которых включает описание (*commit message*). Эти описания несут ценную информацию о типе внесенных изменений, что позволяет классифицировать коммиты по категориям, таким как «исправление ошибки» (*bugfix*), «новая функциональность» (*feature*) и т.д. Такая классификация может быть использована в дальнейшем для анализа качества кода, планирования релизов, оптимизации процессов разработки.

Проблема заключается в том, что ручная классификация трудоемка и не масштабируема, поэтому автоматизированная классификация коммитов представляет собой актуальную задачу. В статье описан модуль автоматической классификации коммитов на основе моделей обучения с поддержкой русского и английского языков.

Исследования в области автоматической классификации коммитов активно развиваются. В работе [2] предложен подход на основе наивного байесовского классификатора для анализа сообщений коммитов, достигший точности 70…80 % на тестовых данных. Авторы использовали нейронные сети для классификации типов изменений, включая анализ кода и метаданных. Большинство работ используют статические модели, обученные на общедоступных датасетах, что снижает точность для специфических доменов.

В статье описывается методология активного обучение *ML*-модели для классификации коммитов. Практическое применение данной методологии заключается в автоматизации процесса сортировки и анализа коммитов в системах контроля версий, что позволяет ускорить выявление важных изменений, повысить точность классификации. Полученные результаты и сбор других метрик в совокупности позволяют определить профиль разработчика, узнать его сильные и слабые стороны. Для наглядности и оценки результатов разработанной методологии разработана программа с веб-интерфейсом для классификации коммитов из реальных *Git*-репозиториев. Основная особенность заключается в интеграции пользовательского одобрения или коррекции классификаций, что позволяет модели адаптироваться и улучшаться на основе обратной связи.

Классификация коммитов по категориям

Система классифицирует коммиты по 11-ти категориям, которые охватывают основные типы изменений в программных проектах. Эти категории соответствуют стандартам разработки ПО [3]. При необходимости и в зависимости от специфики компании можно изменять словарь классификаций. В табл. 1 приведены категории коммитов, которые используются в разработанной программе.

Таблица 1
Table 1

Классификатор категорий коммитов
Classifier of commit categories

№ п/п	Категория коммита (английский язык)	Категория коммита (русский язык)
1	bugfix	исправление ошибок
2	feature	разработка новой функциональности
3	performance	оптимизации производительности системы
4	log	изменения в логировании
5	security	устранение уязвимостей, связанных с безопасностью
6	database	проведение работ с базой данных
7	merge	слияние веток
8	functional fix	исправление функциональных замечаний
9	ui	модификации пользовательского интерфейса
10	config	изменения в конфигурации
11	other	прочие изменения

Каждая категория ассоциирована с набором ключевых слов или фраз, которые могут присутствовать в описании коммитов [4]. В категорию «прочие изменения» попадают те коммиты, которые не удалось классифицировать по другим категориям. Автоматическое распознавание основано на обученной модели, которая присваивает коммиту наиболее

вероятную категорию на основе совпадений с заранее подготовленным словарем.

Описание методологии

Классификация коммитов в *Git*-репозитории имеет следующие особенности: основная информация содержится в сообщении коммитов (*commit message*), при этом сообщения коммитов состоят обычно из 1…3 предложений и могут относиться к разным категориям. Также важны не только наличие, но и частота употребления слов.

В системе для классификации коммитов по категориям применяется алгоритм *Multinominal Naive Bayes* (*MNB*) [5]. Такой алгоритм учитывает частоту появления терминов в документе, а также условную независимость признаков, что хорошо работает для коротких текстов [6]. Алгоритм имеет линейную сложность $O(n \cdot m)$, где n – количество документов, m – количество признаков, хорошо масштабируется на больших репозиториях с тысячами коммитов. В случае с многоклассовой классификацией (до 11-ти категорий) *MNB* демонстрирует хорошую производительность на относительно небольших датасетах (от десятков, до нескольких сотен образцов) [7 – 8].

Ниже приведены результаты сравнительного анализа алгоритма *MNB* с другими аналогами по следующим критериям сравнения: качество на коротких текстах, учет частоты термов, скорость обучения, скорость предсказания, интерпретируемость, устойчивость к шумам.

В исследовании «*A Comparative Study of Text Classification Algorithms*» сравнивались алгоритмы *MNB*, *Logistic Regression*, *SVM* (линейный), *Random Forest* по критерию «качество на коротких текста». В результате на датасете коротких текстов со средней длиной 10…20 слов алгоритм *MNB* показывает результаты, сопоставимые с логистической регрессией, а *SVM* при этом значительно быстрее [9].

При сравнении скорости обучения и предсказания в исследовании «*Scalable Text Classification: A Benchmark*» сделан вывод о том, что *MNB* обучается в 15…20 раз быстрее *SVM* и *Random Forest* на 100 000 коротких текстах [10].

Модель *MNB* нативно работает со счётными признаками – количество вхождений слов, что хорошо для используемой в работе *TF-IDF* векторизации. Вероятность класса описывается по формуле (1):

$$P(y|x) \propto P(y) \cdot \prod_{i=1}^{n} P(x_i \mid y), \quad (1)$$

где x_i – частота i-го слова, y – класс коммита.

Исследование показало, что модель *MNB* на 10…15 % более устойчива к шуму, т.е. к опечаткам и редким словам, чем логистическая регрессия из-за наличия сглаживания Лапласа, которое уменьшает влияние редких термов [11].

Качество модели классификации коммитов проводится с использованием стандартных метрик многоклассовой классификации. Это позволяет количественно измерить эффективность модели на тестовых данных. Оценка производится в два этапа: расчет метрик для каждого класса, с последующим анализом *confusion matrix* для выявления ошибок. Такая матрица представляет собой таблицу, где строки – истинные метрики, столбцы – предсказанные. Для оценки качества в системе используются следующие метрики:

– метрика общей точности (*accuracy*), которая показывает долю правильно классифицируемых примеров среди всех. Данная метрика дает понять, насколько модель правильна, независимо от классов. Метрика интуитивно понятна, но обманчива при дисбалансе классов: если модель всегда предсказывает частый класс, то метрика может быть высокой, но бесполезной;

– метрика точности для конкретного класса k (*precision*) – показывает долю правильно предсказанных положительных среди всех предсказанных как k. Метрика фокусируется на качестве предсказаний – сколько из того, что модель назвала классом k, действительно им является. Высокая метрика *precision* означает малое количество ложных срабатываний;

– метрика полноты для класса k (*recall*) – показывает долю правильно предсказанных, среди всех истинных k. Низкий показатель указывает на то, что модель делает много упущений;

– метрика уверенности (*confidence*) – показывает на сколько модель уверена в предсказании категории.

Программная реализация методологии

Программная реализация методологии для классификации коммитов в *Git*-репозиториях разработана на языке *Python* и состоит из трех основных модулей, каждый из которых выполняет свои функции.

Модуль *connection_to_repository* использует библиотеку *GitPython* и отвечает за подключение к *Git*-репозиторию. Данный модуль имеет следующий принцип работы:

1) выполняется парсинг исходного введенного пользователем *URL* репозитория;
2) встраивание имени пользователя и токена доступа в доменную часть *URL* для того, чтобы подключиться к защищенному репозиторию и формирование на основе полученных данных *URL* вида: «*https://username:token@github.com/owner/repo.git*»;
3) клонирование репозитория и создание временной директории;
4) получение информации о ветках;
5) сохранение состояния в сессии;
6) использование репозитория для дальнейшего анализа.

Процесс подключения к репозиторию и получения входных данных (коммитов) для дальнейшей работы наглядно представляет собой простую стандартную форму ввода *URL git*-репозитория, к которому нужно подключиться, ввести имя пользователя репозитория и его токен. Коммиты система берет из выбранной ветки за указанный пользователем период.

Модуль *commit_category_analyzer* отвечает за обработку текстовых описаний коммитов, проведение морфологического анализа и расширения словаря ключевых слов. Данный модуль включает функции для нормализации текста сообщения, извлечения базовых форм слов и генерации дополнительных словоформ для повышения точности распознавания. Нормализация сообщений коммитов включает в себя приведение к нижнему регистру, удалению специальных символов, чтобы оставались только буквы, цифры и пробелы, а также нормализацию пробелов и табуляций. Для обработки сообщений коммитов на русском языке интегрирован морфологический анализ с использование библиотеки *pymography*2. Такой подход позволяет генерировать словоформы, учитывая грамматические категории, такие как падежи, числа, времена для повышения точности распознавания синонимов и вариаций слов.

Модуль *commit_categoty_ml* реализует машинно-обучаемую модель для классификации коммитов на основе подготовленных данных. Для начала проводится векторизация текста, которая представляет собой процесс преобразования текстовых описаний коммитов в числовые векторы, пригодные для обучения моделей машинного обучения. В системе применяется *TF-IDF* (*Term Frequency-Inverse Document Frequency*) векторизатор, который учитывает как частоту термина в документе (*TF*), так и его редкость во всей коллекции (*IDF*). Это позволяет эффективно выделять информативные слова и фразы, минимизируя влияние распространенных слов типа «изменение» или «добавление», которые могут оказаться шумом.

Параметры векторизатора выбраны эмпирически на основе анализа тестовых данных:

– *max_features* = 5000: ограничивает количество наиболее информативных признаков до 5000, что помогает избежать переобучения на больших словарях и ускоряет обучение. Анализ показал, что для типичных сообщений коммитов (короткие тексты) этого достаточно для покрытия 95 % уникальных терминов;

– *ngram_range* = (1, 3): включает униграммы (отдельные слова), биграммы (парные последовательности) и триграммы (троичные). Это обоснованно необходимостью захватывать фразы по типу «исправление ошибки» или «оптимизация производительности», которые лучше описывают категории, чем отдельные слова;

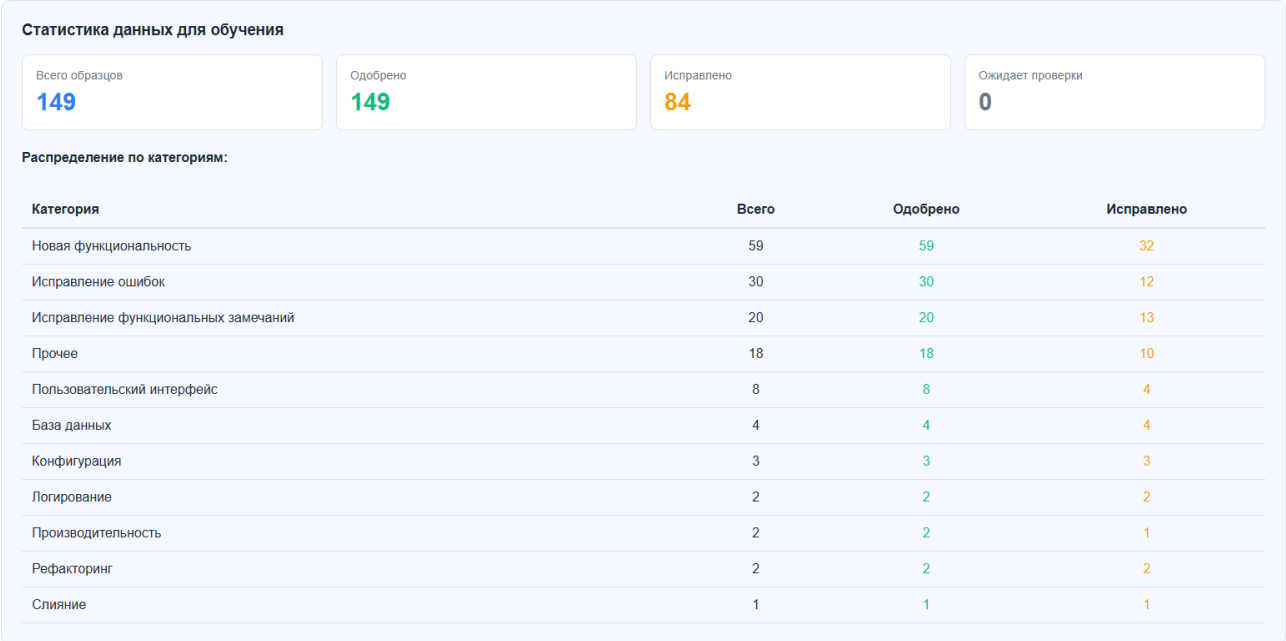
– *min_df* = 1: минимальная частота документа для термина – 1, что позволяет учитывать редкие термины (например, «рефакторинг»), характерные для определенных категорий;

– *max_df* = 0,95: максимальная частота – 95 % от общего числа документов, что фильтрует слишком распространенные термины;

– *stop_words* = None: стоп-слова не удаляются, поскольку сообщения коммитов короткие и могут содержать значимые слова (например, «не» или «нет»), которые теряются при стандартной фильтрации. В отличие от анализа длинных текстов, это повышает точность на небольших данных.

Экспериментальные результаты

Для обучения модели берутся данные о коммитах, взятых из репозиториев, к которым подключается программа. В разделе обучения модели пользователю предоставляется возможность одобрить классификацию, которая была предложена моделью, либо отклонить и изменить классификацию вручную, если она неправильная. Одобренные классификации автоматически становятся образцами для последующего обучения модели после того, как было собрано достаточное количество образцов для обучения модели (минимум 10 экземпляров для работы, рекомендуемое не менее 50-ти разноклассовых).
Созданная модель была обучена на 149-ти образцах, которые представляют собой коммиты в ветке одного из взятых *Git*-репозиториев. На рис. 1 представлена разбивка коммитов по категориям, по которым они распределились в рамках обучения модели.



Статистика данных для обучения

Всего образцов	Одобрено	Исправлено	Ожидает проверки
149	149	84	0

Распределение по категориям:

Категория	Всего	Одобрено	Исправлено
Новая функциональность	59	59	32
Исправление ошибок	30	30	12
Исправление функциональных замечаний	20	20	13
Прочее	18	18	10
Пользовательский интерфейс	8	8	4
База данных	4	4	4
Конфигурация	3	3	3
Логирование	2	2	2
Производительность	2	2	1
Рефакторинг	2	2	2
Слияние	1	1	1

Рис. 1. Обучение модели
Fig. 1. Model training

На рис. 2 представлен процесс обучения модели с помощью активного обучения. Каждый образец, взятый для обучения модели, прошел ручное одобрение в соответствии с правильностью классификации, подобранной моделью. В случае некорректного первоначального подбора пользователь самостоятельно вносит изменения в выбранную классификацию, указывая, модели на ошибку и подсказывает правильную классификацию коммита.

После обучения модели была взята другая ветка того же репозитория и взяты 65 последних коммитов для анализа. В результате при тестировании можно увидеть часть присвоенных моделью категорий коммитов и уверенность модели в этом решении (табл. 2).

Проведенная экспериментальная оценка модели демонстрирует высокую уверенность в классификации коммитов с четко выраженной семантикой. Это говорит о том, что даже при ограниченном объеме обучающих данных *MNB* способен выявлять семантические маркеры, характерные для типов коммитов в разработке ПО. Однако были зафиксированы снижения уверенности в случаях, когда при описании коммитов были использованы неинформативные формулировки, либо наблюдалась неоднозначность между категориями.

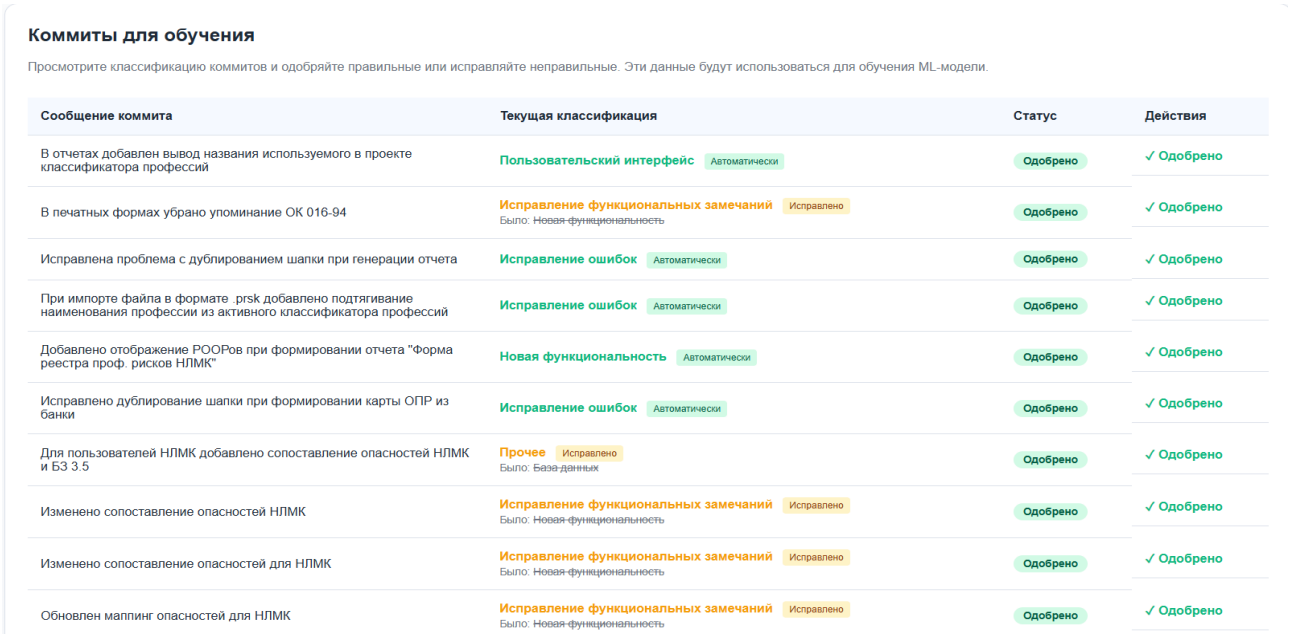


Рис. 2. Процесс активного обучения модели
Fig. 2. The process of actively learning the model

Таблица 2
Table 2

Результат присвоения категории коммита по его сообщению
The result of assigning a commit category based on its message

Сообщение	Категория	Уверенность модели
В отчетах добавлен вывод названия используемого в проекте классификатора профессий	Пользовательский интерфейс	0,95
Исправлена проблема с дублированием шапки при генерации отчета	Исправление ошибок	0,95
В печатных формах убрано упоминание ОК 016-94	Исправление функциональных замечаний	0,95
Исправлено дублирование шапки при формировании карты ОПР из банки	Исправление ошибок	0,95
Update 2026-01-14 *ProfessionMapper.sql*	База данных	0,95
Add new RiskProf.KB.Prof.Client	Новая функциональность	0,516258821
Рекомендуемые при подборе профессии упорядочены по убыванию схожести с необходимой	Исправление ошибок	0,60178998
Update RiskWorkplaceService	Новая функциональность	0,60178998

Заключение

В статье предложена методология активного обучения *ML*-модели для классификации коммитов *Git*-репозитория и ее программная реализация в виде автоматизированной системы, которая позволяет определять категории полученных коммитов, для последующего использования при проведении работ по анализу кода, планированию релизов. В системе используется *ML*-модель, которая обучается пользователем в полуавтоматическом режиме путем одобрения или отклонения классификации коммитов, которые предложила *ML*-модель.

Практический вывод: для внедрения автоматической классификации коммитов в командной разработке необходимо сочетать *ML*-модель и *CI*-проверки, чтобы гарантировать, что тексты коммитов содержат достаточно семантическую информацию о надежности классификации. В таком случае *ML*-подход станет инструментом повышения качества и прозрачности разработки.

Список источников:

1. Крупкин С.А. Работа с системой контроля версий Git [Текст]. – М.: Изд-во Московского университета, 2022. – 120 с.

2. Automated commit classification for git repositories using machine learning technique [Текст] / X. Wang, Y. Jiang, Y. Xu et al. // Proceedings of the 30th

References:

1. Krupkin SA. Working with the Git Version Control System. Moscow: Moscow University Press; 2022.

2. Wang X, Jiang Y, Xu Y, et al. Automated Commit Classification for Git Repositories Using Machine Learning Technique. In: Proceedings of the

ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023). – 2023. – P. 112-124.

3. Конвенция коммитов (Conventional Commits) [Электронный ресурс] / Conventional Commits Initiative. – Version 1.0.0, 2019.

4. Иванов Н.Н. Синтаксический разбор предложения для векторизации текста // Вопросы науки и образования. – 2017. – №11 (12). – С. 45-46.

5. Zhang H., Jiang L., Yu H.-K. A literature review on naive Bayes classifiers // Intelligent Data Analysis. – 2020. – Vol. 24. – No. 1. – P. 37-57.

6. Гусев П.Ю. Обработка текстов и подготовка моделей векторизации для программного комплекса классификации научных текстов // Моделирование, оптимизация и информационные технологии. – 2021. – №9 (1).

7. Terentyeva Yu. Sentiment Analysis, InSet Lexicon, SentiStrength Lexicon, Naive Bayes, Multinomial Naive Bayes, TF-IDF, Machine Learning // International Journal of Open Information Technologies. – 2024. – №7. URL: https://cyberleninka.ru/article/n/sentiment-analysis-inset-lexicon-sentistrength-lexicon-naive-bayes-multinomial-naive-bayes-tf-idf-machine-learning (дата обращения: 10.01.2026).

8. Pascarella L. On the Use of Machine Learning Techniques for Software Engineering Tasks: A Systematic Literature Review // IEEE Transactions on Software Engineering. – 2021. – Т. 47, № 11. – С. 2301-2325.

9. Zhang Y., Wang H., Liu Z. A Comparative Study of Text Classification Algorithms // Journal of Machine Learning Research. – 2018. – Vol. 19. – P. 1-35.

10. Chen M., Li X., Zhou J. Scalable Text Classification: A Benchmark // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL). – 2020. – P. 4567-4579.

11. Wang T., Jiang L., Chen R. Noise-Robust Text Classification with Naive Bayes // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). – 2019. – P. 1234-1243.

30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023): 2023. p. 112-124.

3. Conventional Commits. Conventional Commits Initiative. Version 1.0.0 [Internet]. 2019.

4. Ivanov N.N. Syntactic Parsing of a Sentence for Text Vectorization. Problems of Science and Education. 2017;11:45-46.

5. Zhang H., Jiang L., Yu H-K. A Literature Review on Naive Bayes Classifiers. Intelligent Data Analysis. 2020;24(1):37-57.

6. Gusev PY. Word Processing and Preparation of Vectorization Models for a Software Package for the Classification of Scientific Texts. Modelling, Optimization and Information Technology. 2021;9(1).

7. Terentyeva Yu. Sentiment Analysis, InSet Lexicon, SentiStrength Lexicon, Naive Bayes, Multinomial Naive Bayes, TF-IDF, Machine Learning. International Journal of Open Information Technologies [Internet]. 2024 [cited 2026 Jan 10];7. Available from: https://cyberleninka.ru/article/n/sentiment-analysis-inset-lexicon-sentistrength-lexicon-naive-bayes-multinomial-naive-bayes-tf-idf-machine-learning

8. Pascarella L. On the Use of Machine Learning Techniques for Software Engineering Tasks: A Systematic Literature Review. IEEE Transactions on Software Engineering. 2021;47(11):2301-2325.

9. Zhang Y., Wang H., Liu Z. A Comparative Study of Text Classification Algorithms. Journal of Machine Learning Research. 2018;19:1-35.

10. Chen M, Li X, Zhou J. Scalable Text Classification: A Benchmark. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL): 2020. p. 4567-4579.

11. Wang T, Jiang L, Chen R. Noise-Robust Text Classification with Naive Bayes. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): 2019. p. 1234-1243.

Информация об авторах:

Копелиович Дмитрий Игоревич
кандидат технических наук, заведующий кафедры «Информатика и программное обеспечение» Брянского государственного технического университета, AuthorID-РИНЦ: 350384, ORCID: 0000-0003-4095-7029.

Кургуз Михаил Андреевич
аспирант кафедры «Компьютерные технологии и системы» Брянского государственного технического университета, ORCID: 0009-0008-1270-8918.

Information about the authors:

Kopeliovich Dmitry Igorevich
Candidate of Technical Sciences, Head of the Computer Science and Software Department of Bryansk State Technical University, AuthorID: 350384, ORCID: 0000-0003-4095-7029.

Kurguz Mikhail Andreevich
Postgraduate Student of the Computer Technologies and Systems Department of Bryansk State Technical University, ORCID: 0009-0008-1270-8918.

Вклад авторов: все авторы сделали эквивалентный вклад в подготовку публикации.
Contribution of the authors: the authors contributed equally to this article.

Авторы заявляют об отсутствии конфликта интересов.
The authors declare no conflicts of interests.

Статья поступила в редакцию 24.03.2026; одобрена после рецензирования 15.04.2026; принята к публикации 20.04.2026.

The article was submitted 24.03.2026; approved after reviewing 15.04.2026; accepted for publication 20.04.2026.