Инженерный вестник Дона, №10 (2025)
ivdon.ru/ru/magazine/archive/n10y2025/10448

Использование локального подхода к иерархической классификации текстов

А.А. Сайгин, С.А. Федосин

Мордовский государственный университет им. Н.П. Огарёва, Саранск, Россия

Аннотация: В статье формируется задача иерархической классификации текстов, описываются подходы к иерархической классификации и метрики оценки их работы, подробно рассматривается локальный подход к иерархической классификации, описываются разные подходы к локальной иерархической классификации, проводится серия экспериментов по обучению локальных иерархических классификаторов с различными методами векторизации, сравниваются результаты оценки работы обученных классификаторов.
Ключевые слова: классификация, иерархическая классификация, локальная классификация, иерархическая точность, иерархическая полнота, иерархическая F-мера, обработка естественного языка, векторизация.

Иерархическая классификация – это подвид задачи классификации, то есть задачи разделения объектов в соответствии с определённым критерием, в которой классы организованы в единую иерархическую структуру [1]. В подобной структуре между классами можно построить отношения, в которых одни классы будут являться подклассами других. Визуально иерархию можно представить в виде графа или дерева. Наличие иерархии усложняет задачу классификации, так как результатом могут быть как конечные (листовые) классы, так и промежуточные (внутренние) вершины. Поэтому при решении задачи классификации важно учитывать существующие между классами отношения [2].

Для решения данной задачи разрабатываются различные методы, которые способны автоматически определять принадлежность данных к определенному классу. В настоящее время существует несколько подходов к решению задачи.

Одним из таких подходов является локальный. Данный подход работает «сверху-вниз», то есть сначала выбирается класс на самом верхнем уровне иерархии, следующий класс выбирается среди дочерних, и так до тех

пор, пока не будет выбран листовой класс иерархии. Необходимо обучить несколько классификаторов, каждый из которых будет работать на своем уровне. Таким образом, учитывается только локальный контекст объекта, то есть его связи с ближайшими соседями. Иерархический классификатор представляет собой объединенный набор плоских классификаторов. Преимуществами данного подхода являются эффективное использование отношений между классами, гибкость за счет легкой расширяемости классификатора, простота интерпретации результатов. К недостаткам относят большой размер и сложную архитектуру системы, долгий и ресурсозатратный процесс обучения [3].

Существуют различные подходы к созданию локальных классификаторов. Рассмотрим их на примере иерархии, изображенной на рис. 1.

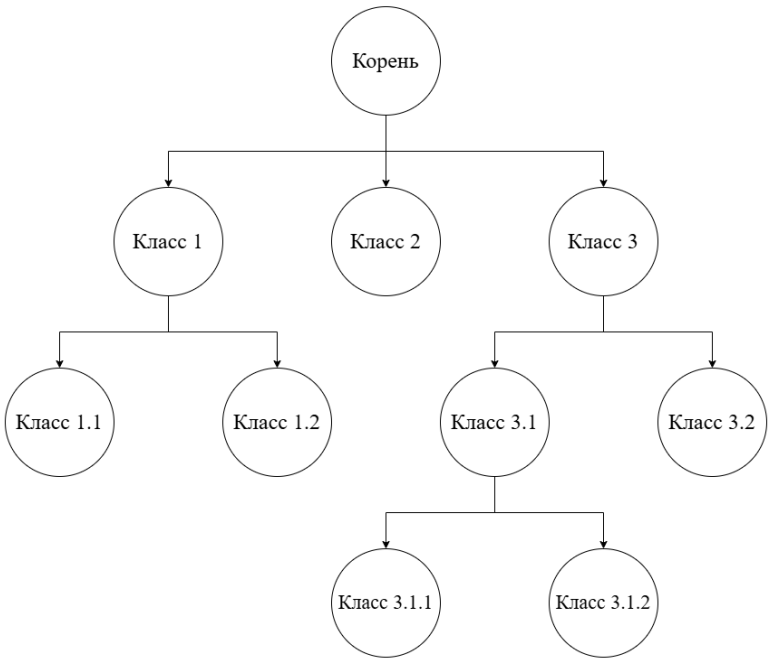

Рис. 1. – Иерархия классов

Первый подход – локальная классификация для вершины (Local Classifier per Node, LCN). В данном подходе в каждом узле создается отдельный классификатор. Каждый классификатор определяет, принадлежит

объект к текущему узлу или нет. В результате получается дерево бинарных классификаторов [3]. Каждый классификатор обучается независимо, что делает подход гибким и легко адаптируемым. Схема работы подхода изображена на рис. 2.

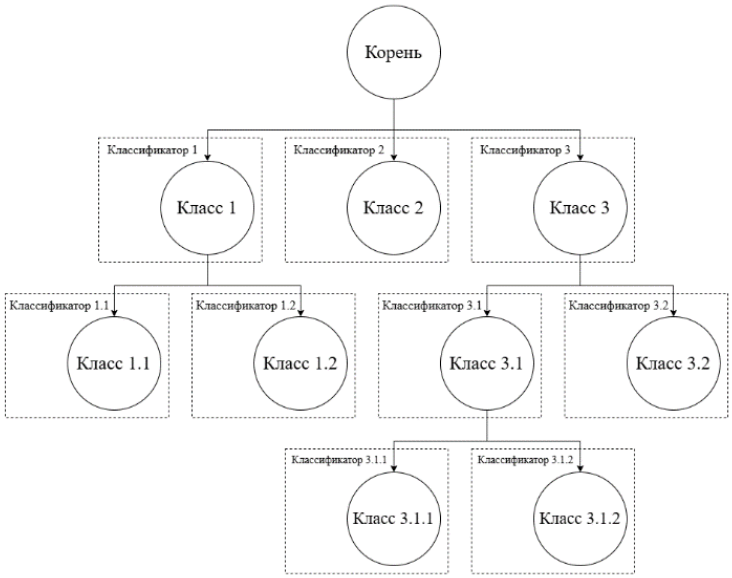

Рис. 2. – Локальная классификация для вершины

Второй подход – локальная классификация для родительской вершины (Local Classifier per Parent Node, LCPN). Как и в LCN, в данном подходе в узлах иерархии находятся классификаторы. Отличие в том, что классификаторы создаются для каждого родительского узла, каждый из которых определяет дочерний узел по отношению к текущему. То есть, в каждом узле находится алгоритм многоклассовой классификации [3]. Как и в предыдущем подходе, каждый классификатор обучается независимо. Схема работы подхода изображена на рис. 3.

Третий подход – локальная классификация для уровня (Local Classifier per Level, LCL). В данном подходе создаются классификаторы для каждого уровня иерархии. Следующий класс определяется с использованием результата, полученного на предыдущем уровне [3]. Схема работы подхода изображена на рис. 4.

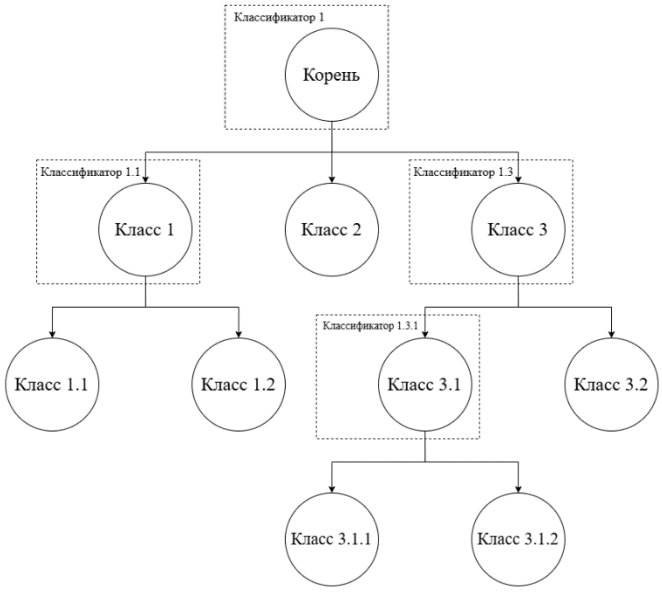

Рис. 3. – Локальная классификация для родительской вершины

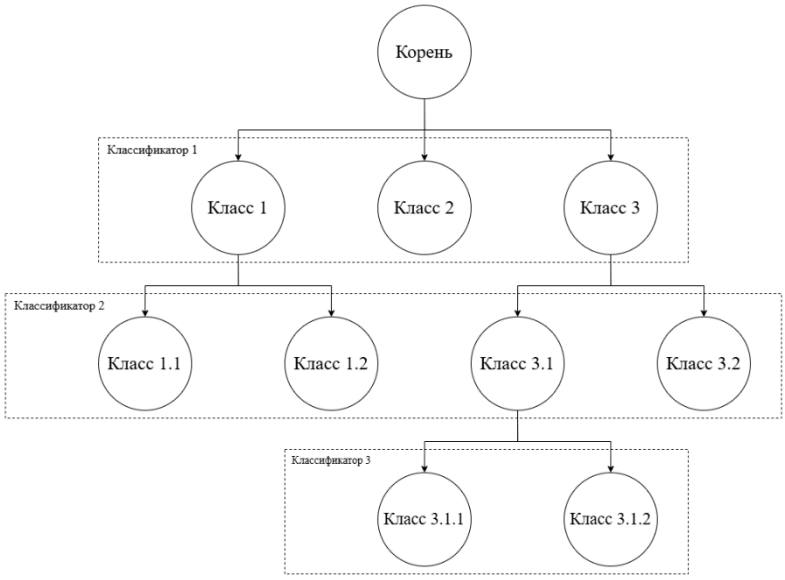

Рис. 4. – Локальная классификация для уровня

Рассмотрим описанные подходы на практике. Для этого обучим локальные классификаторы для каждого из подходов. Примерами иерархически распределенных данных могут быть справочники, каталоги товаров, номенклатуры, статьи. Зачастую данные представлены в текстовом виде, что усложняет их классификацию. Компьютер оперирует числовыми значениями, а не символьными, поэтому текст предварительно нужно

преобразовать в числовые векторы. Алгоритм классификации состоит из этапов предобработки текста, векторизации и классификации.

Для преобразования текстов в числовые векторы разработаны разные алгоритмы векторизации, в том числе на основе методов машинного обучения [4]. Используем некоторые из них для обучения локальных классификаторов, а именно:

- FastText (модель cc.en.300) [5];
- USE (модель google/universal-sentence-encoder) [6];
- BERT (модель google-bert/bert-base-multilingual-cased) [7].

FastText – метод векторного представления слов, использующий нейронную сеть прямого распространения для предсказания слов. Алгоритм использует символьные n-граммы, на которые разбивается текст. N-граммы представляют из себя композицию нескольких последовательностей символов определенной длины. Благодаря разбиению на n-граммы FastText способен обрабатывать редкие и неизвестные слова [5].

Universal Sentence Encoder (USE) – это нейросетевая модель, которая преобразует предложения в векторные представления. USE основана на глубокой нейронной сети, включающей последовательные слои двунаправленного кодера и декодера с механизмом внимания. Модель обучена на обширных текстовых данных, таких как книги, новостные статьи, страницы Википедии и другие источники информации [6].

BERT (Bidirectional Encoder Representations from Transformers) – это модель для представления текста, построенная на основе нейронной сети, базирующейся на архитектуре Transformer. Данная архитектура используют механизм внимания для обработки данных. BERT представляет собой набор последовательно соединённых энкодеров Transformer. Модель обрабатывает входную последовательность слов, передавая её через каждый слой энкодера. На каждом этапе применяется механизм внутреннего внимания, который

выделяет наиболее значимые слова, а затем данные проходят через полносвязную нейронную сеть и передаются следующему энкодеру. Это позволяет модели эффективно учитывать контекст и выделять ключевые семантические элементы текста [7].

Каждая модель имеет свое количество выходов, от чего зависит размер итогового вектора. У выбранных моделей следующее количество выходов: у FastText – 300, USE – 512, BERT – 768.

На каждый уровень будет обучаться своя модель плоской классификации. В рамках эксперимента использовалась классическая нейронная сеть прямого распространения. Это архитектура, которая аппроксимирует функцию с использованием линейных слоёв. В этих слоях входные данные умножаются на веса связей, а затем применяются функции активации в нейронах. Полученные сети включают входной слой, где количество нейронов соответствует выбранному методу векторизации, один скрытый слой с 512 нейронами и функцией активации ReLU, и выходной слой, где число нейронов равно количеству определяемых классов. В LCN у моделей будет два выхода, поскольку решается задача бинарной классификации, у LCPN количество выходов равно количеству дочерних узлов по отношению к текущему, у LCL – количеству классов на текущем уровне. На выходе модели используется функция LogSoftmax. Для обучения модели применяется функция потерь CrossEntropyLoss и оптимизатор AdamW. Обучение каждой модели проводилось в течение 100 эпох.

Обучение модели проводилось на датасете DBPedia Classes, который представляет собой структурированную информацию, извлечённую из Википедии. Датасет включает 342782 записи, организованные в трёхуровневую иерархию, где количество классов на каждом уровне составляет 9, 70 и 219 соответственно [8]. На рис. 5 изображены диаграммы распределения данных по классам на каждом уровне иерархии.

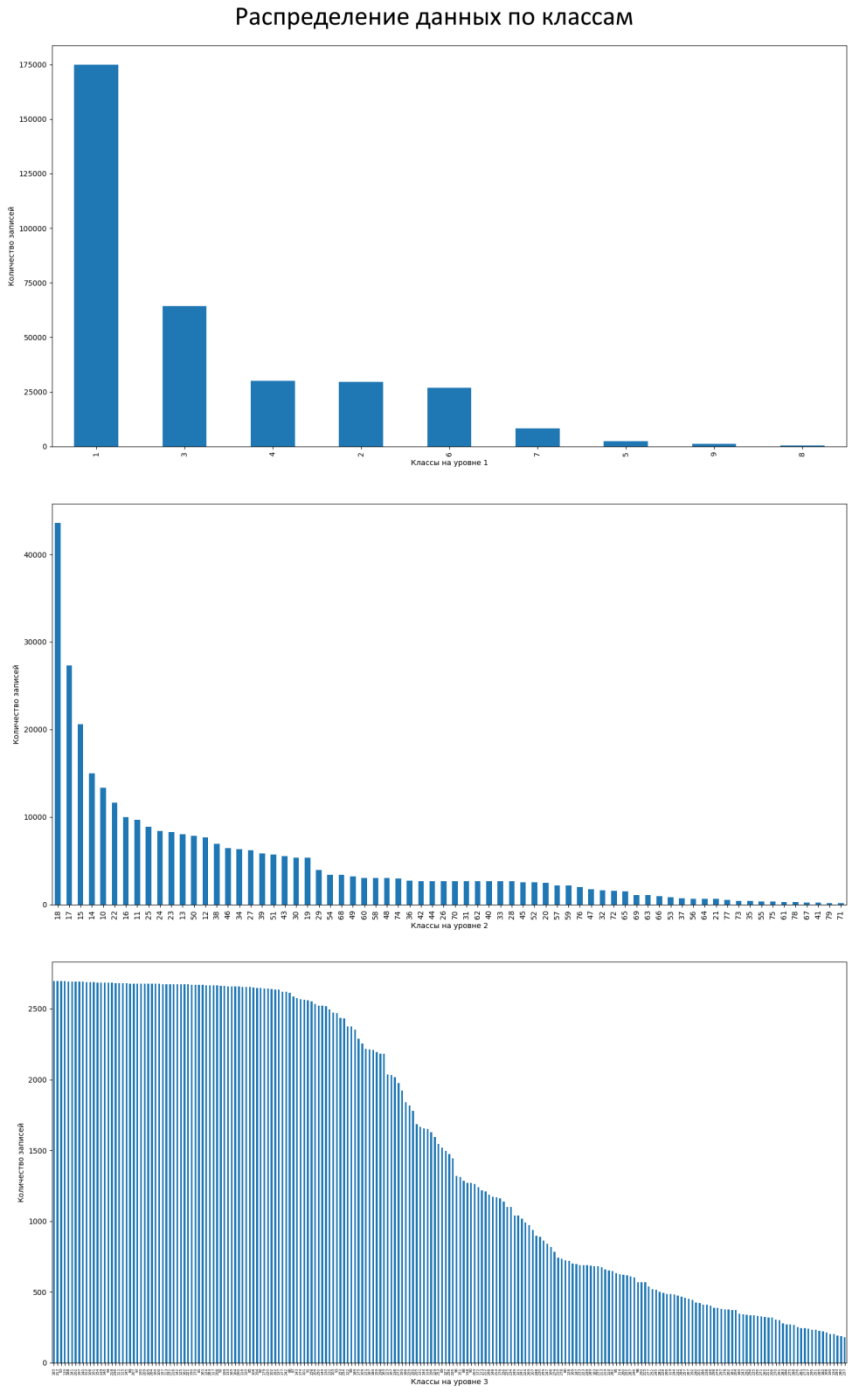

Рис. 5. – Распределение данных по классам

Схема получившегося плоского классификатора изображена на рис. 6. Схема изображает архитектуру модели, которая используется в локальной классификации по уровням на первом уровне иерархии и обучалась на векторах google-bert/bert-base-multilingual-cased. Количество входов в данном

примере равно 768, что соответствует выходу модели BERT. Количество выходов равно 9, поскольку столько классов находится на первом уровне иерархии выбранного набора данных. Архитектуры были разработана с помощью фреймворка PyTorch языка программирования Python [9].

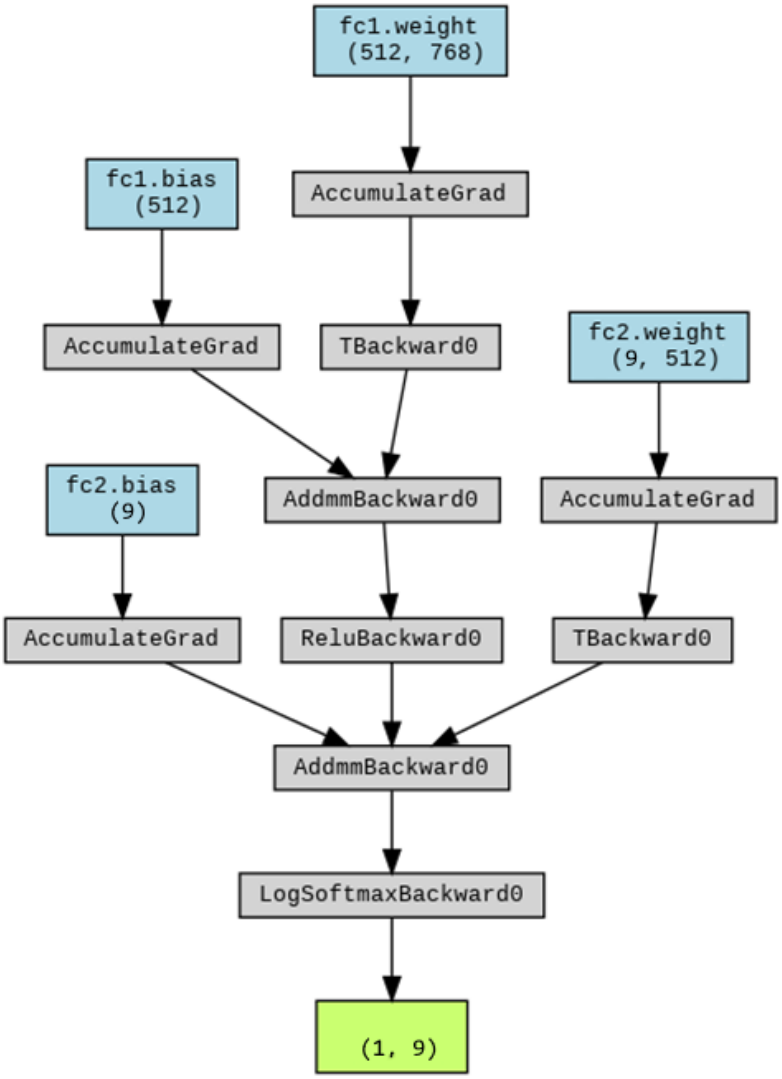

Рис. 6. – Пример архитектуры классификатора, использующегося на конкретном уровне иерархии

Для оценки модели классификации обычно используются метрики точности, полноты и F-меры. Но данные метрики оценивают только точность определения конечного класса. Возможны ситуации, когда итоговый класс предсказан неверно, но результат имеет общих предков с истинным классом. Эти предсказания также нужно учитывать, для этого существуют отдельные

метрики. Иерархическая точность – это отношение числа общих предков предсказанного и истинного классов к числу предков истинного класса. Иерархическая полнота – это отношение числа общих предков предсказанного и истинного классов к числу предков прогнозируемого класса. Иерархическая F-мера – это единая оценка иерархической классификации, она равна среднему гармоническому между иерархической точностью и иерархической полнотой. Обозначим множество истинных классов объекта как $A(C_t)$, а множество предсказанных классов объекта как $A(C_p)$. Тогда иерархическая точность будет вычисляться по формуле (1), иерархическая полнота – по формуле (2), иерархическая F-мера – по формуле (3) [10].

$$hR = \frac{|A(C_t) \cap A(C_p)|}{A(C_t)}. \quad (1)$$

$$hP = \frac{|A(C_t) \cap A(C_p)|}{A(C_p)}. \quad (2)$$

$$hF = \frac{2 * hP * hR}{hP + hR}. \quad (3)$$

Используем для оценки иерархическую F-меру и сравним ее с обычной F-мерой.

Результаты обучения моделей представлены в таблице 1.

Таблица №1

Результаты обучения классификаторов

Классификатор	FastText		USE		BERT	
	F	hF	F	hF	F	hf
LCN	0,89	0,92	0,89	0,93	0,89	0,93
LCPN	0,92	0,94	0,90	0,93	0,90	0,94
LCL	0,90	0,93	0,90	0,93	0,89	0,93

Результаты обучения показали высокие результаты классификации. По F-мере видна высокая точность итоговой классификации, иерархическая F-мера показывает, что у части объектов были верно определены вышестоящие в иерархии классы. Все три подхода показали примерно одинаковые результаты. Возникает проблема, что в разных подходах используется разное количество моделей, из-за чего увеличивается время обучения. В нашем примере в классификации для вершин обучалось 298 моделей, в классификации для родительских вершин – 80 моделей, а в классификации для уровней – 3 модели. С другой стороны, гибкость локального подхода позволяет использовать разные алгоритмы машинного обучения в рамках одного классификатора. Например, на верхних уровнях иерархии, где вариативность входных данных шире, можно использовать нейронные сети, а на нижних уровнях, где набор поступающих данных уже, можно обучить линейный или байесовский классификатор. За счет этого можно регулировать точность результата, время обучения и размер итогового набора моделей.

Литература

1. Silla C. N., Freitas A. A. A survey of hierarchical classification across different application domains // Data mining and knowledge discovery. 2011. V. 22. pp. 31-72.

2. Денискин А. В., Федосин С. А., Плотникова Н. П. Применение иерархической кластеризации DIANA для улучшения качества классификации текста // Инженерный вестник Дона. 2023. № 9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8662

3. Borges H. B., Silla Jr C. N., Nievola J. C. An evaluation of global-model hierarchical classification algorithms for hierarchical classification problems with single path of labels // Computers & Mathematics with Applications. 2013. V. 66. №. 10. pp. 1991-2002.

4. Сайгин А. А., Федосин С. А. Использование метода определения схожести слов для оценки алгоритмов векторизации // Инженерный вестник Дона. 2024. № 7. URL: ivdon.ru/ru/magazine/archive/n7y2024/9381.

5. Grave E., Bojanowski P., Gupta P., Joulin A., Mikolov T. Learning word vectors for 157 languages // arXiv preprint arXiv:1802.06893. – 2018. URL: arxiv.org/abs/1802.06893 (дата обращения: 26 июля 2025).

6. Cer D., Yang Y., Kong S. Y., Hua N., Limtiaco N., John R. S., Constant N., Guajardo-Cespedes M., Yuan S., Tar C., Sung Y.-H., Strope B., Kurzweil R. Universal sentence encoder // arXiv preprint arXiv:1803.11175. – 2018. URL: arxiv.org/abs/1803.11175 (дата обращения: 26 июля 2025).

7. Devlin J., Chang M. W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. – 2018. URL: arxiv.org/abs/1810.04805 (дата обращения: 26 июля 2025).

8. DBPedia Classes: Kaggle: сайт – 2010. URL: kaggle.com/datasets/danofer/dbpedia-classes (дата обращения: 26.07.2025).

9. PyTorch: сайт – 2016. URL : pytorch.org (дата обращения: 26.07.2025).

10. Kiritchenko S., Matwin S., Nock R., Famil A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization // Conference of the Canadian society for computational studies of intelligence. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. pp. 395-406.

References

1. Silla C. N., Freitas A. A. Data mining and knowledge discovery. 2011. T. 22. pp. 31-72.

2. Deniskin A. V., Fedosin S. A., Plotnikova N. P. Inzhenernyi vestnik Dona. 2023. № 9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8662

3. Borges H. B., Silla Jr C. N., Nievola J. C. Computers & Mathematics with Applications. 2013. T. 66. №. 10. pp. 1991-2002.

4. Saygin A. A., Fedosin S. A. Inzhenernyi vestnik Dona. 2024. № 7. URL: ivdon.ru/ru/magazine/archive/n7y2024/9381.

5. Grave E., Bojanowski P., Gupta P., Joulin A., Mikolov T. arXiv preprint arXiv:1802.06893. – 2018. URL: arxiv.org/abs/1802.06893 (Date accessed: 26 iulya 2025).

6. Cer D., Yang Y., Kong S. Y., Hua N., Limtiaco N., John R. S., Constant N., Guajardo-Cespedes M., Yuan S., Tar C., Sung Y.-H., Strope B., Kurzweil R. arXiv preprint arXiv:1803.11175. 2018. URL: arxiv.org/abs/1803.11175 (Date accessed: 26 iulya 2025).

7. Devlin J., Chang M. W., Lee K., Toutanova K. arXiv preprint arXiv:1810.04805. 2018. URL: arxiv.org/abs/1810.04805 (Date accessed: 26 iulya 2025).

8. DBPedia Classes: Kaggle: sajt 2010. URL: kaggle.com/ datasets/danofer/dbpedia-classes (Date accessed: 26.07.2025).

9. PyTorch : sajt – 2016. URL : pytorch.org (Date accessed: 26.07.2025)

10. Kiritchenko S., Matwin S., Nock R., Famil A. F. Conference of the Canadian society for computational studies of intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. pp. 395-406.

Дата поступления: 5.08.2025

Дата публикации: 25.09.2025