Информатика. Экономика. Управление//
Informatics. Economics. Management

2023; 2(4)
http://oajiem.com/

УДК: 004.85
DOI: https://doi.org/10.47813/2782-5280-2023-2-4-0119-0133

EDN: **URBCCF**

# Использование нейронных сетей глубокого обучения для классификации токсичных комментариев в социальных сетях

**Д. В. Захаренко**

*Сибирский федеральный университет, Институт космических и информационных технологий, Красноярск, Россия*

**Аннотация.** Целью этого исследования было изучение использования искусственных нейронных сетей глубокого обучения для классификации токсичных комментариев в социальных. Распространенность токсичных взаимодействий на этих платформах достигла небывало высокого уровня, что привело к снижению уровня цифровой цивилизованности. Модераторы этих платформ вынуждены тратить большое количество времени и сил, чтобы контролировать негатив в комментариях. В исследовании рассматриваются различные алгоритмы и методы построения искусственных нейронных сетей, а также сравнивается производительность трех выбранных моделей, чтобы определить наиболее эффективную для решения этой задачи. Комментарии со страницы обсуждения в Википедии выполняют роль данных для построения моделей классификации. Исследование включает в себя обзор методов, используемых для достижения целевых результатов, с использованием Python и его библиотек. Оно также охватывает технические аспекты, такие как процесс построения, обучения и оценки моделей искусственных нейронных сетей. Была рассмотрена ценная информация о необходимых теоретических основах, а также обсуждены некоторые предыдущие исследования и решения. Классификация характера комментариев, содержащих ненависть, обеспечит платформам гибкость в работе с ними и откроет двери для новых обсуждений и решений.

**Ключевые слова:** искусственные нейронные сети, глубокое обучение, классификация текста, предобработка текста, токсичные комментарии, социальные сети, цифровая цивилизованность.

**Для цитирования:** Захаренко, Д. В. (2023). Использование нейронных сетей глубокого обучения для классификации токсичных комментариев в социальных сетях. Информатика. Экономика. Управление - Informatics. Economics. Management, 2(4), 0119–0133. https://doi.org/10.47813/2782-5280-2023-2-4-0119-0133

# Using deep learning neural networks to classify toxic comments on social media

**D. V. Zakharenko**

*Siberian Federal University, Institute of Space and Information Technologies, Krasnoyarsk, Russia*

**Abstract.** The purpose of this study was to study the use of artificial neural networks of deep learning to classify toxic comments on social networks. The prevalence of toxic interactions on these platforms has reached an all-time high level, which has led to a decrease in the level of digital civility. Moderators of these platforms have to spend a lot of time and effort to control the negative in the comments. The study examines various algorithms and methods for building artificial neural networks, and compares the performance of the three selected models to determine the most effective for solving this problem. Comments from the Wikipedia discussion page serve as data for building classification models. The study includes an overview of the methods used to achieve targeted results using Python and its libraries. It also covers technical aspects, such as the process of building, training and evaluating models of artificial neural networks. Valuable information about the necessary theoretical foundations was reviewed, as well as some previous studies and solutions were discussed. Classifying the nature of hate comments will provide platforms with flexibility in dealing with them and open the door to new discussions and solutions.

**Keywords:** artificial neural networks, deep learning, text classification, text preprocessing, toxic comments, social networks, digital civility.

**For citation:** Zakharenko, D. V. (2023). Using deep learning neural networks to classify toxic comments on social media. Informatics. Economics. Management, 2(4), 0119–0133. https://doi.org/10.47813/2782-5280-2023-2-4-0119-0133

## ВВЕДЕНИЕ

В наше время большое количество людей общаются друг с другом в комментариях к публикациям в социальных сетях. Возможность комментировать позволяет людям открыто выражать свои мысли и чувства, а также облегчает обмен информацией и помогает людям в поиске решений различных проблем.

Согласно индексу цифровой цивилизованности Microsoft, вежливость онлайн-общения достигла четырехлетнего минимума в 2019 году [1]. Токсичные взаимодействия происходят на платформах социальных сетей, где преобладают жестокие политические дебаты и нападки на личностные качества. То, что должно было быть хорошей социальной силой, платформой для свободного обсуждения в Интернете, превратилось в трясину ненависти. В этом контексте машинное обучение может оказать огромную

помощь, поскольку то, на что у человека могут уйти часы работы, может быть выполнено искусственным интеллектом за несколько секунд.

В статье рассматриваются различные алгоритмы и методы построения искусственных нейронных сетей, а также сравнивается производительность трех выбранных моделей, чтобы определить наиболее эффективную для решения этой задачи. Комментарии со страницы обсуждения в Википедии выполняют роль данных для построения моделей классификации.

Статья включает в себя обзор методов, используемых для достижения целевых результатов, с использованием Python и его библиотек. Она также охватывает технические аспекты, такие как процесс построения, обучения и оценки моделей искусственных нейронных сетей. Была рассмотрена ценная информация о необходимых теоретических основах, а также обсуждены некоторые предыдущие исследования и решения. Классификация характера комментариев, содержащих ненависть, обеспечит платформам гибкость в работе с ними и откроет двери для новых обсуждений и решений.

## МАТЕРИАЛЫ И МЕТОДЫ

### Набор данных

Был использован набор данных, взятый с интернет-платформы Kaggle, содержащий комментарии со страницы обсуждения в Википедии. Набор содержит 2 типа комментариев: токсичные и нормальные. Токсичные комментарии могут быть отнесены к разным классам: токсичные, сильно токсичные, непристойные, содержащие угрозу, оскорбительные, содержащие ненависть к личности.

Тестовый набор данных делится на 2 части. Первая публичная часть содержит примерно 10% объема данных, вторая приватная часть содержит остальные 90% данных.

### Предварительная обработка данных

Предварительная обработка данных - важнейший этап машинного обучения, который повышает качество данных, способствует извлечению из них ценной информации и преобразует необработанные данные в формат, который является читаемым и понятным [2].

Подготовка необработанных данных может включать в себя множество шагов, таких как очистка данных и обработка пропущенных значений. Количество шагов

предварительной обработки может варьироваться в зависимости от количества дефектов в данных.

В разделах ниже представлены все этапы предварительной обработки, предпринятые для подготовки данных к обучению.

*Очистка данных*

Наиболее распространенным методом очистки текста является приведение к заглавным или строчным буквам из-за разнообразия заглавных букв для формирования предложений. Этот метод позволяет спроецировать все слова в тексте и документе в одно и то же пространство объектов.

Следующим шагом является трансформация сокращений в полные словосочетания.

Далее необходимо удалить различные ненужные символов, такие как излишние знаки препинания, URL-адреса и символы, отличные от ASCII.

*Токенизация и индексация*

Токенизация является важным методом обработки естественного языка, при котором текст разбивается на более мелкие фрагменты, называемые токенами. Эти токены могут состоять из любого отдельного слова, фразы или целого предложения и используются в качестве входных данных для моделей машинного обучения. Процесс токенизации включает в себя удаление всего синтаксиса, специальных символов и пробелов из текста перед разделением его на набор токенов, поскольку это позволит модели машинного обучения анализировать текст более детальным образом, а затем определять шаблоны и взаимосвязи между токенами [3].

Индексация – это метод присвоения числовых индексов категориальным переменным или объектам в наборе данных. Поскольку большинство алгоритмов машинного обучения требуют числовых данных в качестве входных данных, этот метод имеет решающее значение. Легче оценивать и анализировать категориальные переменные, когда они преобразуются в числовые значения с использованием индексации [4].

Все шаги, упомянутые выше, могут быть выполнены с помощью класса Tokenizer, который является частью библиотеки Keras для языка программирования Python. Класс дает возможность преобразовать каждый фрагмент текста либо в последовательность

целых чисел, где каждое целое число соответствует индексу токена в предопределенном словаре, либо в вектор, где каждый токен представлен двоичным коэффициентом, полученным из количества слов.

*Лемматизация*

Лемматизация – это процесс, при котором слова преобразуются в корневую форму, сохраняя при этом их значение в контексте фразы. Лемматизацию часто противопоставляют стеммингу, другому методу разбиения слов на их простейшие формы, хотя он менее грубый и сохраняет первоначальное значение текста нетронутым [5].

В этом исследовании был использован класс WordNetLemmatizer из библиотеки ntkl. Этот класс работает путем определения части речи слова перед использованием лексической базы данных WordNet для поиска базовой формы или леммы.

*Стоп-слова*

Стоп-слова – это распространенные слова естественного языка, которые часто удаляются из текстовых данных во время предварительной обработки. Эти термины лишены значимости и потенциально могут внести шум в анализ, что приведет к неточным моделям. Такие слова, как "the", "and", "an" и "in", являются примерами стоп-слов.

Удаление стоп-слов может повлиять на качество и точность модели за счет уменьшения размерности информации и сосредоточения внимания на более значимых словах. Однако важно иметь в виду, что удаление слишком большого количества стоп-слов может привести к потере некоторой контекстуальной информации, что приведет к получению менее точных моделей [6].

В этом исследовании используется библиотека spacy, поскольку в ней имеется обширный список стоп-слов. После тщательного изучения было замечено, что данные содержали ненужные случаи, когда однобуквенные или двухбуквенные слова существовали без какого-либо смысла. Чтобы преодолеть эту проблему, к списку по умолчанию, предоставляемому библиотекой spacy, был добавлен пользовательский список стоп-слов, который содержал одиночные и двойные буквы от a до z (например, aa, bc). Некоторые буквы были удалены из пользовательского списка, такие как me, am, as, поскольку они придают важное значение предложению.

*Заполнение*

Комментарии из онлайн-беседы имеют переменную длину. Некоторые могут состоять из одного слова, в то время как другие могут быть длиной в абзац. Алгоритмы машинного обучения не могут передавать в модель данные несогласованной длины, чтобы обойти эту проблему, вводится заполнение.

Заполнение может быть введено в начало или конец последовательности. Нули добавляются к последовательности до тех пор, пока она не достигнет требуемой длины [7].

Для реализации заполнения в этом исследовании была использована функция pad_sequences из библиотеки Keras. Для всех предложений короче 200 слов в конце вектора последовательности были добавлены нули.

**Разделение набора данных на обучение, тестирование и проверку**

После прохождения этапов предварительной обработки данные более стабильны и согласованы, что приводит к следующему шагу - разделению выборки с помощью функции train_test_split из библиотеки sklearn.

Набор данных делится на 3 части: обучение, тестирование и валидация. Каждый из разделов данных имеет отличный от другого вариант использования и играет значительную роль в процессе машинного обучения.

**Обучение и тестирование моделей**

В этом разделе рассматриваются 3 модели, созданные для обучения, и приводится соответствующая важная информация о различных параметрах, которые можно учитывать при создании модели машинного обучения.

Все 3 модели имеют разные архитектуры. Первая – модель LSTM с 8 слоями, вторая – модель CNN с 8 слоями, третья – гибридная модель LSTM-CNN с 12 слоями.

*LSTM*

Первой реализованной моделью стала искусственная нейронная сеть с долгой краткосрочной памятью, поскольку она наиболее согласуется с текстовыми данными. Иерархическое отображение модели LSTM и информация о слоях и нейронах отображены на рисунке 1.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150)]             0

 embedding (Embedding)       (None, 150, 300)          30000000

 lstm_layer (LSTM)           (None, 150, 50)           70200

 global_max_pooling1d (Glob  (None, 50)                0
 alMaxPooling1D)

 dropout (Dropout)           (None, 50)                0

 dense (Dense)               (None, 40)                2040

 dropout_1 (Dropout)         (None, 40)                0

 dense_1 (Dense)             (None, 6)                 246

=================================================================
Total params: 30072486 (114.72 MB)
Trainable params: 72486 (283.15 KB)
Non-trainable params: 30000000 (114.44 MB)
_________________________________________________________________
```

Рисунок 1. Архитектура модели LSTM.

Figure 1. Architecture of the LSTM model.

Модель состоит из одного входного слоя, одного embedding слоя, за которым следует один слой LSTM с 50 нейронами. Модель имеет 2 слоя дропаут, 2 полносвязных слоя и один слой подвыборки в порядке, показанном на рисунке 1. Общее количество параметров составило 30 072 486, из которых 72 486 были поддающимися обучению.

Дропаут слой (dropout) – это метод регуляризации, используемый в нейронных сетях для предотвращения переобучения. Во время обучения некоторые нейроны в слоях отсева случайным образом устанавливаются равными нулю, предотвращая чрезмерную зависимость модели от какой-либо одной функции или набора функций. Это способствует повышению эффективности обобщения модели [8].

Полносвязный слой – это основной строительный элемент нейронной сети. Каждый нейрон в одном слое соединен с каждым нейроном в следующем. Полносвязные слои могут быть многослойными для построения глубоких нейронных сетей и используются для изучения сложных нелинейных корреляций между входными и выходными данными [9].

*CNN*

Второй созданной моделью была одномерная сверточная нейронная сеть. Иерархическое отображение модели CNN изображено на рисунке 2.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 150)]             0

 embedding_1 (Embedding)     (None, 150, 300)          30000000

 conv1d (Conv1D)             (None, 146, 128)          192128

 max_pooling1d (MaxPooling1  (None, 29, 128)           0
 D)

 flatten (Flatten)           (None, 3712)              0

 dropout_2 (Dropout)         (None, 3712)              0

 dense_2 (Dense)             (None, 128)               475264

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 30668166 (116.99 MB)
Trainable params: 668166 (2.55 MB)
Non-trainable params: 30000000 (114.44 MB)
_________________________________________________________________
```

Рисунок 2. Архитектура модели CNN.

Figure 2. Architecture of the CNN model.

Модель состоит из одного входного слоя и одного embedding слоя, за которым следует один слой свертки. Модель имеет 1 дропаут слой, 2 полносвязных слоя, 1 слой подвыборки и один слой flatten в порядке, показанном на рисунке 2. Общее количество параметров составило 30 668 166, из которых 668 166 были поддающимися обучению.

Сверточные нейронные сети (CNN) – это тип нейронных сетей, который доказал свою исключительную эффективность в приложениях распознавания изображений. Однако он был успешно применен в приложениях для обработки естественного языка (NLP), таких как классификация текста. CNN отлично подходит для категоризации текста, поскольку он может распознавать основные свойства текста, такие как n-граммы и фразы, не требуя сложной разработки функций.

Архитектура CNN состоит из последовательности сверточных слоев, которые сканируют входную матрицу на наличие шаблонов или объектов с помощью фильтров. Фильтры применяются к каждому возможному окну ввода слов, позволяя сети изучать шаблоны в различных масштабах. Выходные данные сверточного слоя впоследствии

передаются через слой подвыборки, который уменьшает размерность и собирает наиболее значимую информацию. Наконец, прежде чем принять окончательное решение о классификации, выходные данные слоя подвыборки передаются через один или несколько полносвязных слоев [10].

*LSTM-CNN*

Третьей созданной моделью был гибрид долговременной краткосрочной памяти и одномерной сверточной нейронной сети. Иерархическое отображение модели LSTM-CNN и информацию о слоях и нейронах можно увидеть на рисунке 3.

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 150)]             0

 embedding_2 (Embedding)     (None, 150, 300)          30000000

 lstm_layer (LSTM)           (None, 150, 50)           70200

 conv1d_1 (Conv1D)           (None, 150, 64)           9664

 max_pooling1d_1 (MaxPoolin  (None, 50, 64)            0
 g1D)

 global_max_pooling1d_1 (Gl  (None, 64)                0
 obalMaxPooling1D)

 batch_normalization (Batch  (None, 64)                256
 Normalization)

 dense_4 (Dense)             (None, 40)                2600

 dropout_3 (Dropout)         (None, 40)                0

 dense_5 (Dense)             (None, 30)                1230

 dropout_4 (Dropout)         (None, 30)                0

 dense_6 (Dense)             (None, 6)                 186

=================================================================
Total params: 30084136 (114.76 MB)
Trainable params: 84008 (328.16 KB)
Non-trainable params: 30000128 (114.44 MB)
_________________________________________________________________
```

Рисунок 3. Архитектура модели LSTM-CNN.

Figure 3. Architecture of the LSTM-CNN model.

Модель состоит из одного входного слоя, одного embedding слоя, за которым следует один слой LSTM с 50 нейронами и один слой свертки. Модель имеет 2 слоя дропаут, 3 полносвязных слоя, 2 слоя подвыборки и один слой пакетной нормализации в порядке, показанном на рисунке 3. Общее количество параметров составило 30 084 136, из которых 84 008 были поддающимися обучению.

## РЕЗУЛЬТАТЫ

Для модели LSTM были достигнуты следующие результаты. Потери при обучении 0.0478, потери при валидации 0.0461, точность при обучении 0.9922, точность при валидации 0.9926. Рисунок 4 отображает изменение величин точности и потерь во время обучения модели.

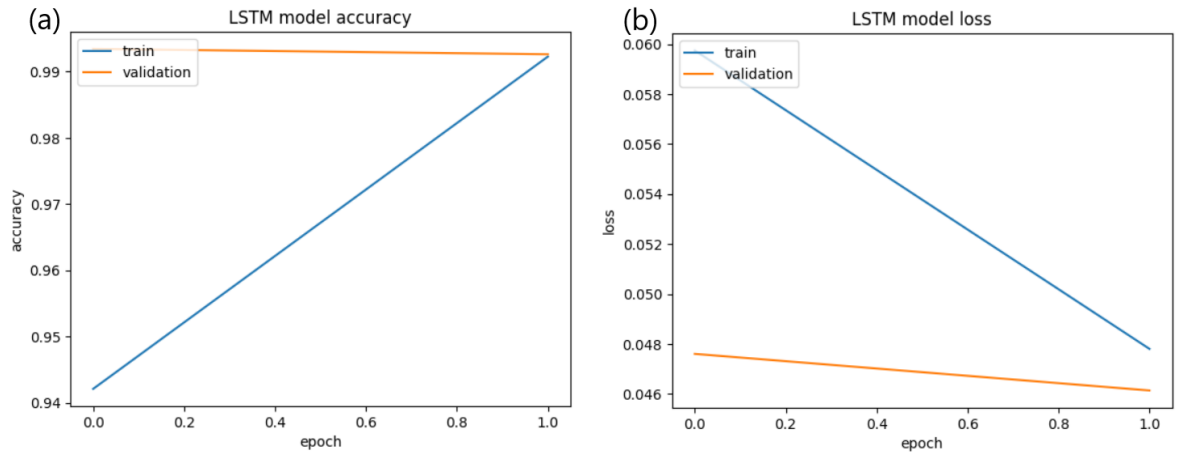


Рисунок 4. (a) Точность модели LSTM; (b) Потери модели LSTM.

Figure 4. (a) LSTM model accuracy; (b) LSTM model loss.

Оценка точности LSTM модели на публичном тестовом наборе данных равна 0.97327, а на приватном 0.97402. (рисунок 5)

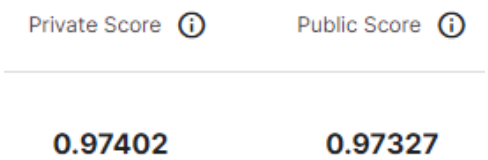


Рисунок 5. Точность модели LSTM на тестовом наборе данных.

Figure 5. LSTM model accuracy on the test dataset.

Для модели CNN были достигнуты следующие результаты. Потери при обучении 0.0550, потери при валидации 0.0566, точность при обучении 0.9664, точность при валидации 0.9891. Рисунок 6 отображает изменение величин точности и потерь во время обучения модели.

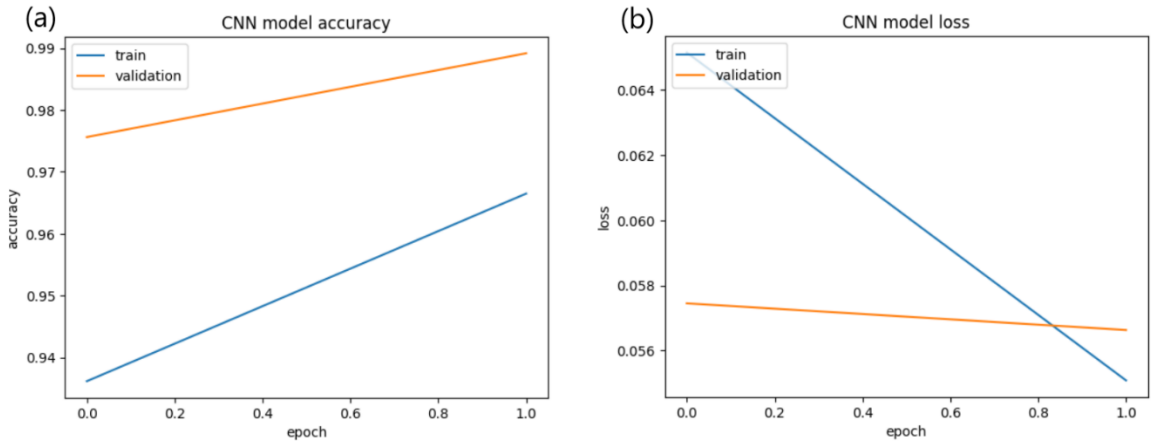


Рисунок 6. (a) Точность модели CNN; (b) Потери модели CNN.

Figure 6. (a) CNN model accuracy; (b) CNN model loss.

Оценка точности CNN модели на публичном тестовом наборе данных равна 0.95133, а на приватном 0.95273. (рисунок 7)


Рисунок 7. Точность модели CNN на тестовом наборе данных.

Figure 7. CNN model accuracy on the test dataset.

Для модели LSTM-CNN были достигнуты следующие результаты. Потери при обучении 0.0490, потери при валидации 0.0469, точность при обучении 0.9898, точность при валидации 0.9932. Рисунок 8 отображает изменение величин точности и потерь во время обучения модели.

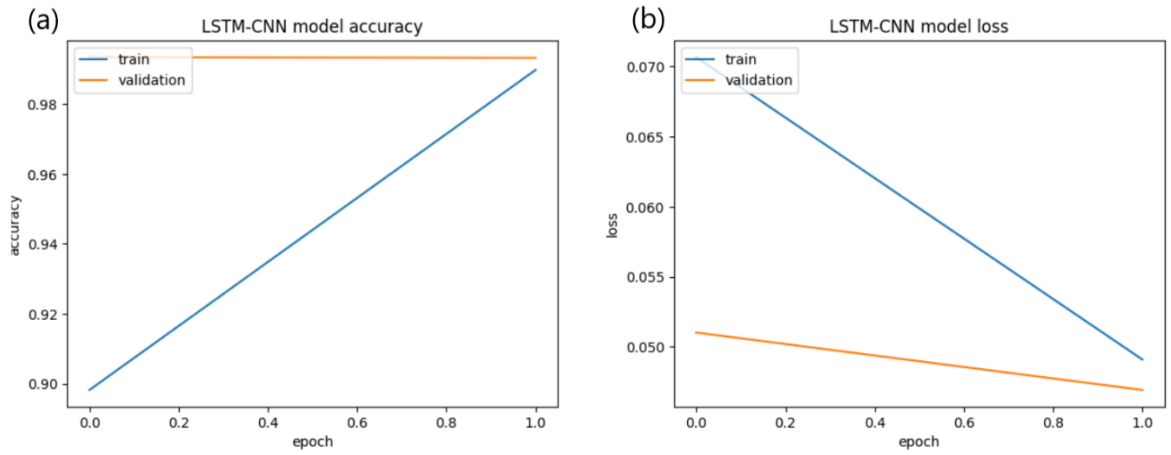


Рисунок 8. (a) Точность модели LSTM-CNN; (b) Потери модели LSTM-CNN.

Figure 8. (a) LSTM-CNN model accuracy; (b) LSTM-CNN model loss.

Оценка точности LSTM-CNN модели на публичном тестовом наборе данных равна 0.97851, а на приватном 0.97648. (рисунок 9)

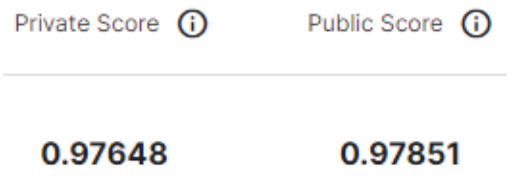


Рисунок 9. Точность модели LSTM-CNN на тестовом наборе данных.

Figure 9. LSTM-CNN model accuracy on the test dataset.

## ОБСУЖДЕНИЕ

Для LSTM модели были опробованы самые различные методы настройки. Использование слоев подвыборки и дропаут, а также изменение функции оптимизатора и функции потерь дало отличный результат.

Модель CNN уступает другим моделям по величине точности, однако все еще показывает хороший результат.

Гибридная модель LSTM-CNN содержала больше слоев по сравнению с остальными моделями. Внедрение слоев подвыборки и слоев дропаут оказали хорошее влияние на результат. Данная модель показала лучшую точность.

Производительность для всех трех моделей глубокого обучения была значительной. Есть некоторые области, которые могут быть улучшены с целью повышения показателей точности.

Все модели были настроены по-разному друг от друга, и выбранные гиперпараметры были основаны на методе проб и ошибок. Использование передовых методов, таких как поиск по сетке, может значительно помочь в подборе гиперпараметров.

Еще одна область, которую можно улучшить, – это список стоп-слов. Составление списка стоп-слов, связанных с используемым набором данных, может помочь повысить производительность.

## ЗАКЛЮЧЕНИЕ

В результате в рамках статьи был выполнен обзор существующих исследований, который показал, что сверточные нейронные сети, рекуррентные нейронные сети и их гибрид являются одними из наиболее эффективных моделей для обнаружения и классификации токсичных комментариев. Эти модели продемонстрировали высокую точность, высокую скорость обработки и надежную производительность в различных контекстах.

Однако использование глубокого обучения и искусственных нейронных сетей для обнаружения токсичных комментариев также сопряжено с некоторыми трудностями. Например, модели необходимо обучать на разнообразных и репрезентативных наборах данных, чтобы гарантировать то, что они могут эффективно обобщаться на новые данные. Кроме того, модели должны уметь учитывать сложные нюансы языка, включая сарказм, иронию и культурные отсылки.

Будущие исследования должны быть сосредоточены на решении этих проблем и повышении производительности глубокого обучения и искусственных нейронных сетей для обнаружения токсичных комментариев. Они могут включать в себя изучение новых архитектур и оптимизацию гиперпараметров.

Выводы этой статьи предполагают, что глубокое обучение и искусственные нейронные сети обладают большим потенциалом для решения проблемы токсичных комментариев в социальных сетях. Предоставляя точные и эффективные решения для обнаружения и классификации токсичных комментариев, эти модели могут способствовать здоровому и уважительному взаимодействию в Интернете.

## СПИСОК ЛИТЕРАТУРЫ

[1] Data Preprocessing in Machine learning. URL: https://blogs.microsoft.com/on-the-issues/2020/02/10/digital-civility-lowest. (дата обращения: 14.09.2023).

[2] Javatpoint. URL https://www.javatpoint.com/data-preprocessing-machine-learning. (дата обращения: 17.09.2023).

[3] NTKL. nltk.tokenize package — NLTK 3.8.1. URL: https://www.nltk.org/api/nltk.tokenize.html. (дата обращения: 19.09.2023).

[4] Brownlee, J. Why One-Hot Encode Data in Machine Learning?. URL: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning. (дата обращения: 21.09.2023).

[5] WordNet. A Lexical Database for English. URL: https://wordnet.princeton.edu. (дата обращения: 23.09.2023).

[6] Datastart. Плавное введение в Natural Language Processing (NLP). URL: https://datastart.ru/blog/read/plavnoe-vvedenie-v-natural-language-processing-nlp. (дата обращения: 25.09.2023).

[7] Brownlee, J. Data Preparation for Variable Length Input Sequences. URL: https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction. (дата обращения: 27.09.2023).

[8] TensorFlow. tf.keras.layers.Dropout. TensorFlow Core v2.14.0. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout. (дата обращения: 29.09.2023)

[9] TensorFlow. tf.keras.layers.Dense. TensorFlow Core v2.14.0. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense. (дата обращения: 03.10.2023)

[10] Randolph. Deep Learning for Multi-Label Text Classification. URL: https://github.com/RandolphVI/Multi-Label-Text-Classification. (дата обращения: 14.10.2023)

## REFERENCES

[1] Data Preprocessing in Machine learning. URL: https://blogs.microsoft.com/on-the-issues/2020/02/10/digital-civility-lowest. (data obrashcheniya: 14.09.2023).

[2] Javatpoint. URL https://www.javatpoint.com/data-preprocessing-machine-learning. (data obrashcheniya: 17.09.2023).

[3] NTKL. nltk.tokenize package — NLTK 3.8.1. URL: https://www.nltk.org/api/nltk.tokenize.html. (data obrashcheniya: 19.09.2023).

[4] Brownlee, J. Why One-Hot Encode Data in Machine Learning?. URL: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning. (data obrashcheniya: 21.09.2023).

[5] WordNet. A Lexical Database for English. URL: https://wordnet.princeton.edu. (data obrashcheniya: 23.09.2023).

[6] Datastart. Plavnoe vvedenie v Natural Language Processing (NLP). URL: https://datastart.ru/blog/read/plavnoe-vvedenie-v-natural-language-processing-nlp. (data obrashcheniya: 25.09.2023).

[7] Brownlee, J. Data Preparation for Variable Length Input Sequences. URL: https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction. (data obrashcheniya: 27.09.2023).

[8] TensorFlow. tf.keras.layers.Dropout. TensorFlow Core v2.14.0. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout. (data obrashcheniya: 29.09.2023)

[9] TensorFlow. tf.keras.layers.Dense. TensorFlow Core v2.14.0. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense. (data obrashcheniya: 03.10.2023)

[10] Randolph. Deep Learning for Multi-Label Text Classification. URL: https://github.com/RandolphVI/Multi-Label-Text-Classification. (data obrashcheniya: 14.10.2023)

**ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS**

**Захаренко Данил Вадимович,** Сибирский федеральный университет, Институт космических и информационных технологий, кафедра Программной инженерии, Красноярск, Россия
ORCID: 0009-0000-0306-5684

**Danil Zakharenko,** Siberian Federal University, Institute of Space and Information Technologies, Department of Software Engineering, Krasnoyarsk, Russia

*Статья поступила в редакцию 27.10.2023; одобрена после рецензирования 21.11.2023; принята к публикации 22.11.2023.*

*The article was submitted 27.10.2023; approved after reviewing 21.11.2023; accepted for publication 22.11.2023.*