International Journal of Advanced Engineering Research and Science (IJAERS)
Peer-Reviewed Journal
ISSN: 2349-6495(P) | 2456-1908(O)
Vol-13, Issue-2; Feb, 2026
Journal Home Page Available: https://ijaers.com/
Article DOI: https://dx.doi.org/10.22161/ijaers.132.4

Sentiment Classification of Product Reviews

N. S. Fedotov

Financial University under the Government of the Russian Federation

Received: 06 Jan 2026,

Received in revised form: 08 Feb 2026,

Accepted: 12 Feb 2026,

Available online: 19 Feb 2026

***Keywords*— Sentiment analysis, e-commerce, natural language processing, machine learning, BERT, text classification, marketplace analytics**

Abstract *— The rapid growth of e-commerce platforms has led to a significant increase in the volume of user-generated reviews, making manual analysis of customer feedback impractical. A critical issue in modern marketplaces is the presence of inconsistencies between the textual content of reviews and the numerical ratings assigned by users, which can distort product evaluation and mislead potential buyers. This study addresses the task of automatic sentiment classification of textual reviews to identify such discrepancies. A dataset of 100,000 user reviews from the Wildberries marketplace was collected and preprocessed. A three-class sentiment labeling scheme was applied, categorizing reviews as negative, neutral, or positive based on user ratings. Three approaches to sentiment analysis were implemented and compared: logistic regression with TF-IDF vectorization, a transformer-based BERT model, and a hybrid RuBERT model augmented with convolutional neural networks. Experimental results demonstrate that transformer-based and hybrid neural architectures significantly outperform classical machine learning methods, with the RuBERT-CNN model achieving the highest classification performance. The proposed approach can be effectively applied to improve the reliability of rating systems and enhance moderation and analytics tools in e-commerce platforms.*

I. INTRODUCTION

In the context of the rapid growth of e-commerce, marketplaces have become the primary channel of interaction between customers and product manufacturers. User reviews containing textual comments and numerical ratings play a key role in building trust in products, influence purchasing decisions, and directly affect the reputation of sellers and platforms as a whole [1]. At the same time, the volume of generated reviews is constantly increasing, making manual analysis and verification impractical [2].

A significant problem of modern marketplaces is the presence of reviews with inconsistencies between textual content and the assigned rating, for example, a positive comment accompanied by a minimal score or, conversely, a negative text with a high rating [3]. Such anomalous reviews may arise due to various reasons, including accidental rating errors, emotional or sarcastic language, or deliberate manipulation of ratings [4]. The presence of such reviews distorts the overall perception of product quality and may mislead potential customers.

Therefore, the task of automatic sentiment analysis of textual reviews is becoming increasingly relevant. The application of natural language processing and machine learning methods makes it possible to identify discrepancies between review content and assigned ratings, thus ensuring a more objective and reliable assessment of product quality. Consequently, the research and development of sentiment classification methods represent an important and in-demand task for modern e-commerce.

The practical significance of this work lies in the possibility of applying the developed sentiment analysis methods and models in real-world marketplace information systems and e-commerce services. The implementation of an algorithm

capable of automatically determining the sentiment of a textual review and comparing it with the user-assigned rating can be used for preliminary filtering and labeling of incorrect reviews.

This approach improves the quality of user-generated content by identifying erroneous, random, or potentially unreliable ratings without the need to completely remove such reviews from the system. The results of this study can be used to enhance product rating mechanisms, increase user trust in review systems, and provide sellers with more accurate feedback on product quality.

In addition, the proposed approach can serve as a foundation for developing auxiliary tools for moderation, user behavior analytics, and reputation monitoring of products on marketplaces. The developed models can also be adapted for other services, such as booking or delivery platforms, making the research results universal and applicable to a wide range of practical tasks.

II. MATERIALS AND METHODS

The research was conducted at the Department of Artificial Intelligence, Financial University under the Government of the Russian Federation. We used the dataset with info from Wildberries, one of the most famous Russian Russian online retailer[5], and implemented the system in Python with the following libraries: pandas [6], matplotlib [7], scikit-learn [8], networkx [9], torch [10], and numpy [11].

III. RESULTS OF THE RESEARCH

Table 1- Python libraries used for this article

Python libraries
import os
import json
import io
import re
import zstandard **as** zstd
import pandas **as** pd
import numpy **as** np
import matplotlib.pyplot **as** plt
from tqdm **import** tqdm
from sklearn.model_selection **import** train_test_split
from sklearn.linear_model **import** LogisticRegression
from sklearn.feature_extraction.text **import** TfidfVectorizer
from sklearn.pipeline **import** Pipeline
from sklearn.metrics **import** classification_report, accuracy_score, f1_score
from sklearn.utils **import** compute_sample_weight
import torch
import torch.nn **as** nn
from torch.utils.data **import** DataLoader, Dataset
from transformers **import** AutoTokenizer, AutoModel

In this study, the initial data consist of a dataset of user reviews from the Wildberries marketplace, containing textual comments and corresponding numerical product ratings. The data are stored in compressed JSON Lines format, which allows efficient storage and processing of large volumes of information. To ensure experiment reproducibility and limit computational complexity, the dataset size was restricted to the first 100,000 reviews.

	nmId	productValuation	text
0	10001632	4	Для таких же "глазастых" как я, покупателей: н...
1	10001632	5	Отличное средство
2	10001632	5	Дополняю. Запах очень приятный у средства (в п...
3	10001634	5	Отличный! Советую к покупке)
4	10001634	5	Пахнет обалденно

Fig. 1 – Part of the dataset

At the initial stage, data loading and decompression are performed using the Zstandard algorithm. The file is read line by line, each line is converted from JSON format into a Python object and stored in a temporary list. This approach enables streaming data processing without loading the entire dataset into memory, which is crucial when working with large datasets. After reading is complete, the data are transformed into a tabular format using the pandas library for convenient further processing and analysis.

Table 2 – part of the code for reading data from the file

Custom code

path = "data/feedbacks-09.json.zst"
mns = {0:"плохо", 1:"нормально", 2:"хорошо"}
rows = []
limit = 100_000
with open(path, "rb") **as** f:
  dctx = zstd.ZstdDecompressor()
  with dctx.stream_reader(f) **as** reader:
  text_stream = io.TextIOWrapper(reader, encoding="utf-8")
  for i, line **in** enumerate(text_stream):

```
            if i >= limit:
                break
            if not line.strip():
                continue
            rows.append(json.loads(line))
df = pd.DataFrame(rows)
```

The resulting DataFrame contains product identifiers, numerical ratings, review texts, and additional fields not relevant to the task. At this stage, auxiliary columns that do not affect sentiment analysis, such as seller responses and product color information, are removed. As a result, a minimal feature set is retained, including product ID, rating, and review text.

Next, a text preprocessing stage is performed, which is critically important for improving machine learning model quality. Review texts are converted to lowercase to eliminate case-based discrepancies. All characters other than Cyrillic and Latin letters, digits, and spaces are removed, eliminating punctuation, special symbols, and other noise elements. Additional whitespace normalization and trimming are applied. The output of this stage is a new column, clean_text, containing a cleaned and unified version of the original review text.

To formalize the sentiment classification task, a target variable label is introduced. A three-class sentiment scale aligned with user ratings is used. Ratings of 1 and 2 are interpreted as negative sentiment, rating 3 as neutral sentiment, and ratings of 4 and 5 as positive sentiment. This transformation reduces the problem to a multiclass classification task and simplifies result interpretation.

At the final data preparation stage, missing value checks are performed, and the rating column is converted to an integer data type to ensure computational correctness. The final dataset contains 100,000 records with five columns: product ID, original rating, original review text, cleaned text, and sentiment label.

```
import pandas as pd
import re

df['clean_text'] = (df['text'].str.lower().str.replace(r'[^a-za-я0-9\s]', ' ', regex=True).str.replace(r'\s+', ' ', r
df['label'] = df['productValuation'].apply(lambda x: 0 if x in [1,2] else (1 if x==3 else 2))

df = df.dropna(subset=['text'])
df = df[df['text'] != '']

df['productValuation'] = df['productValuation'].astype(int)
df
```

	nmId	productValuation	text	clean_text	lab
0	10001632	4	Для таких же "глазастых" как я, покупателей: н...	для таких же глазастых как я покупателей на па...	
1	10001632	5	Отличное средство	отличное средство	
2	10001632	5	Дополняю. Запах очень приятный у средства (в п...	дополняю запах очень приятный у средства в про...	
3	10001634	5	Отличный! Советую к покупке)	отличный советую к покупке	
4	10001634	5	Пахнет обалденно	пахнет обалденно	

Fig.2 - Data preprocessing

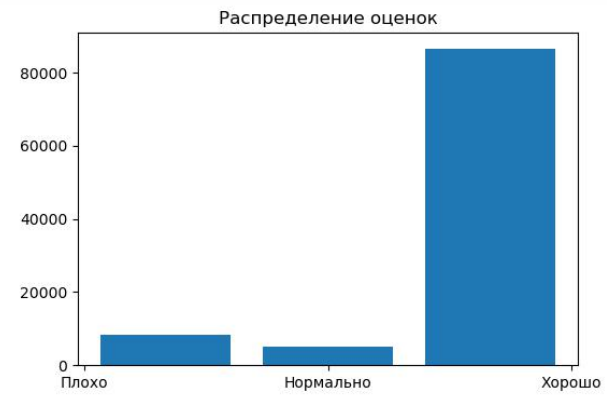

Fig. 3 - Label distribution

Model Development

In the experimental part of the study, three models representing different approaches to text processing were implemented and trained to solve the sentiment classification task: a classical machine learning method based on statistical vectorization, a transformer-based neural network model (BERT), and a hybrid architecture combining RuBERT contextual representations with convolutional neural networks.

Logistic Regression with TF-IDF Vectorization

As a baseline model, logistic regression combined with TF-IDF text vectorization was used. The model was implemented as a pipeline that integrates text vectorization and classification. TF-IDF represents each review as a feature vector reflecting word importance relative to the entire corpus. Logistic regression performs multiclass sentiment classification on these vectors. Despite its simplicity, this method is widely used in sentiment analysis and serves as a reliable baseline.

Table 3 – the code of the logistic regression with TF-IDF vectorization

Custom code
w = compute_sample_weight(class_weight="balanced", y=y_train)
model = Pipeline([('tfidf', TfidfVectorizer()), ('clf', LogisticRegression(max_iter=200))])

This model has shown good results (Fig. 4), but more complex models can improve this result.

```
model.fit(X_train, y_train, clf__sample_weight=w)
pred = model.predict(X_test)
probs = model.predict_proba(X_test)

report = classification_report(y_test, pred, output_dict=True)
acc1 = report["accuracy"]
f11 = report["macro avg"]["f1-score"]
print(f"Accuracy: {acc1} \nF1 macro: {f11}")
```

```
Accuracy: 0.84115
F1 macro: 0.6017875682866666
```

Fig.4 - metrics of the logistic regression model with TF-IDF vectorization

BERT

The second model is based on the pretrained BERT transformer architecture, adapted for Russian-language text processing. A custom dataset class was implemented to tokenize texts and generate fixed-length input tensors. The classifier architecture uses the contextual representation of the [CLS] token extracted from BERT outputs, followed by regularization and a fully connected layer to produce class probabilities. This approach effectively captures contextual dependencies in user reviews.

Table 4 – custom code of the BERT architecture

Custom code
class BertClassifier(nn.Module): **def** __init__(self, model_name="DeepPavlov/rubert-base-cased", num_classes=3): super().__init__() self.bert = AutoModel.from_pretrained(model_name) self.fc = nn.Linear(768, num_classes) self.dropout = nn.Dropout(0.3) **def** forward(self, input_ids, attention_mask): out = self.bert(input_ids=input_ids, attention_mask=attention_mask) cls = out.last_hidden_state[:, 0, :] x = self.dropout(cls) **return** self.fc(x)

This model showed an increase in both metrics(Fig. 5), which indicates that the model was chosen correctly. Next, let's try to improve this architecture.

```
Accuracy: 0.8679
F1 macro: 0.656475361823425
```

Fig.5 - BERT metrics

RuBERT with Convolutional Layers

The third model is a hybrid architecture combining RuBERT embeddings with convolutional neural networks. Token-level embeddings from the last BERT layer are passed through multiple one-dimensional convolutional layers with different kernel sizes to capture local n-gram patterns. The resulting features are pooled, concatenated, regularized, and fed into a fully connected classifier. This approach combines global contextual understanding with effective local feature extraction and demonstrated the best performance.

Table 4 - RuBERT with convolutional layers

Custom code
class RuBertCNN(nn.Module): **def** __init__(self, model_name="DeepPavlov/rubert-base-cased", num_classes=3): super().__init__() self.bert = AutoModel.from_pretrained(model_name) self.convs = nn.ModuleList([nn.Conv1d(in_channels=768, out_channels=128, kernel_size=k) **for** k **in** [3,4,5]]) self.dropout = nn.Dropout(0.3) self.fc = nn.Linear(128 * 3, num_classes) **def** forward(self, input_ids, attention_mask): outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask) x = outputs.last_hidden_state.transpose(1, 2) xs = [torch.relu(conv(x)).max(dim=2)[0] **for** conv **in** self.convs] x = torch.cat(xs, dim=1) x = self.dropout(x) **return** self.fc(x)

This model has shown the best results

```
Accuracy: 0.88595
F1 macro: 0.6645493580019481
```

Fig.6 - Metrics of RuBERT with convolutional layers

	Model	Accuracy	F1 macro
0	Log Reg	0.84115	0.601788
1	BERT	0.86790	0.656475
2	RuBERT+	0.88595	0.664549

Fig. 7 - Architecture comparison

Comparing this architecture with others (Fig. 7), you can see the growth of both metrics, which indicates a well-chosen model. This architecture has shown the best results and its use in practice gives positive results.

IV. CONCLUSION

This study addressed the task of sentiment classification of marketplace user reviews to identify discrepancies between textual content and assigned ratings. A dataset of 100,000 Wildberries reviews was collected and preprocessed, and a three-class sentiment scheme was defined.

Three models were implemented and compared: logistic regression with TF-IDF, RuBERT, and a hybrid RuBERT-CNN model. The results demonstrate that modern neural network architectures significantly outperform classical approaches, particularly under class imbalance conditions. The hybrid model achieved the highest Accuracy and F1 macro scores.

The findings confirm the effectiveness of natural language processing and deep learning methods for analyzing user reviews on marketplaces. Future work may include aspect-based sentiment analysis, sarcasm detection, class balancing techniques, and integration of the developed models into real moderation and analytics systems.

REFERENCES

[1] Anindya Ghose and Panagiotis G. Ipeirotis. Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics. IEEE Transactions on Knowledge and Data Engineering, 23(10), 2011.
[2] Bing Liu. Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 2012.
[3] Xue Han, Zhenzhen Zhao, and Xiaoming Zhang. Inconsistency Detection between Review Text and Rating via Sentiment Analysis. Neural Computing and Applications, 36(4), 2024.
[4] Mohamed Amine Benbrahim, Abdellah Chehri, and Hicham Medromi. Fake and Manipulated Reviews Detection in E-Commerce Using Machine Learning Techniques. Information, 11(2), 2020.
[5] Dataset of data from Wildberries - https://huggingface.co/datasets/nyuuzyou/wb-feedbacks
[6] Pandas - https://pandas.pydata.org/
[7] Matplotlib: Visualization with Pythonhttps://matplotlib.org/
[8] Scikit-learn machine learning in python -https://scikit-learn.org/stable/
[9] NetworkX documentation - https://networkx.org/
[10] PyTorch documentation - https://pytorch.org/
[11] Numpy - https://numpy.org/