DOI: 10.55648/1998-6920-2026-20-1-23-38 УДК 004.89:81’322

Software System for Hybrid Sentiment Analysis of Uzbek Texts using Named Entities

B. R. Saidov[1], V. B. Barakhnin[2,3]

[1]Novosibirsk State University
[2]Federal Research Center for Information and Computational Technologies
[3]Siberian State University of Telecommunications and Information Science (SibSUTIS)

Abstract: This article presents a software package for automatic sentiment analysis of Uzbek texts. The system relies on a hybrid approach, combining a transformer model, a named entity extraction (NER) module, and a specially compiled sentiment dictionary of the Uzbek language. The relevance of this development is due to the growing volume of informal texts on social networks and the lack of ready-made tools for processing them. The package implements a full processing cycle: text cleaning and normalization, entity extraction, sentiment detection and keyword detection, and visualization of the results in a built-in web interface. The models and dictionary are adapted to the agglutinative and orthographic features of the Uzbek language, increasing resilience to colloquial and mixed forms of writing. The package's architecture, main software modules and their interactions, as well as the operating principle of the application interface (REST API) are briefly described. Examples of the system's application for analyzing user reviews and messages are provided, confirming its suitability for applied opinion monitoring tasks. Based on the results of initial experiments, a significant improvement in quality is achieved compared to basic text models without taking into account NER and lexicon.

The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation for Federal Research Center for Information and Computational Technologies.

Keywords: sentiment analysis; Uzbek language; named entity recognition; emotion lexicon; hybrid model; software system.

For citation: Saidov B. R., Barakhnin V. B. Software system for hybrid sentiment analysis of Uzbek texts using named entities [Paper Preparation Manual for Vestnik SibGUTI]. Vestnik SibGUTI, 2026, vol. 20, no. 1, pp. 23–38. https://doi.org/10.55648/1998-6920-2026-20-1-23-38.

The article was submitted: 04.12.2025;
revised version: 15.12.2025;
accepted for publication 05.02.2026.

1. Introduction

Over the past ten to fifteen years, the volume of Uzbek-language texts in open sources has grown significantly [1-3]. Users leave reviews of products and services, discuss news, share opinions on government agencies, educational institutions, transportation infrastructure, and much more. Much of this communication takes place on social media and instant messaging apps, where messages are short, emotional, and often written in mixed spelling [4-6]. For organizations working with Uzbek-speaking audiences, the natural need arises: to quickly understand the general sentiment of users and identify the objects to which this sentiment relates. Manual analysis of such texts is only possible with small samples. With large data sets, this becomes a labor-intensive and poorly

scalable task. Here, it makes sense to apply automated sentiment analysis methods and related approaches from natural language processing [7, 8]. For English and a number of other languages, there are ready-made commercial and open-source solutions that can be integrated into existing information systems practically out-of-the-box. For the Uzbek language, the situation is significantly more complex: specialized tools are few, and many universal models produce unstable results due to the language's specificity and the dominance of other languages in the training data [9].

The peculiarities of the Uzbek language and real-world writing practices pose a particular challenge. Uzbek is an agglutinative language, in which grammatical meanings are often encoded through chains of affixes. Informal communication makes extensive use of abbreviations, colloquial forms, a mixture of Latin and Cyrillic scripts, elements of transliteration, and sprinklings of Russian and English words. All this leads to a large number of spelling variations for the same lexical unit and hinders the direct use of standard methods without adaptation. Moreover, in many applied tasks, it is important not only to determine the overall sentiment of a text (positive, negative, or neutral) but also to understand who or what this assessment refers to. A user may be satisfied with the quality of service but dissatisfied with the prices; they may have a positive attitude toward the city as a whole but negatively toward individual services or companies. Therefore, there is a natural need to combine sentiment analysis and named entity extraction (NER) to associate emotional evaluations with specific organizations, brands, geographic locations, and other meaningful entities [10]. In the author's previous work, we proposed: first, an Uzbek emotional dictionary that takes into account the cultural and linguistic characteristics of emotional expression and evaluation [11]; second, hybrid sentiment analysis models that combine contextual representations based on transformers with information about named entities [12]. Experiments have shown that incorporating NER and a lexicon improves performance compared to models relying solely on text without additional information. However, these results were obtained primarily through computational experiments and were not presented as a complete software solution ready for practical use.

The next logical step is to create a software suite that integrates the developed models and linguistic resources into a single system. Such a system should support the full processing cycle: from text input and normalization to the output of interpretable results in a user-friendly format–with sentiment indication, named entity highlighting, and the ability to integrate with external services. It is important that the system be sufficiently flexible, allowing for the integration of new models, the adaptation of dictionaries, and the customization of the interface for specific application scenarios (review monitoring, social media comment analysis, citizen appeals, etc.).

The goal of this paper is to describe the architecture and implementation of a software system for hybrid sentiment analysis of Uzbek texts using named entities and an emotional dictionary, and to demonstrate its performance on real data. This article formulates system requirements, examines the structure and functions of the main modules, describes the technological solutions used (including a web interface and API), and presents the results of initial experiments. Potential future developments for the system are discussed, including scaling the approach to other Turkic languages and expanding the range of supported text processing tasks.

2. Problem Statement and Software System Requirements

In this paper, we are not simply considering a standalone sentiment analysis model; we want to develop a working tool that can be used in real-world situations. Therefore, the main objective is formulated as follows: to create a software system that accepts Uzbek-language texts and returns a user-friendly result–text sentiment and information about the objects to which this assessment relates. This primarily applies to short messages and reviews: comments on social media, reviews of products and services, messages from citizens to organizations, etc. At the same time, the system should not be tailored to a single domain–we want to use it in both research and applied projects.

2.1. Input and Output

The system receives Uzbek-language text as input. In practice, this could be:

1) individual phrases or short messages;
2) short review texts;
3) dialogue fragments.

Such texts often contain a mixture of Latin and Cyrillic characters, colloquial abbreviations, emoji, and mixed writing etc.

We expect the system to provide the following output:

1) an overall sentiment rating for the text (positive, negative, or neutral, at a minimum);
2) a list of identified named entities (people, organizations, cities, etc.) with their type;
3) if possible, a relationship between the entity and the rating (e.g., "bank X – negative", "city Y – positive");
4) data that is easy to work with in the interface: the positions of entities in the text, highlighting tags, etc.

In other words, the user should see not only "the text is negative", but also who or what is being rated.

2.2. Main System Functions

For such a system to be truly useful, it needs to be able to do several things [13-15]. Let's list them not as a formal list of requirements, but as steps that each text goes through within the system:

- *Preprocessing*. The text needs to be slightly "tidied up": remove technical junk, normalize obvious repeating characters, and carefully handle emoji and mixed-language characters [16].
- *Entity extraction.* Next, the system must identify who or what is being discussed: it could be a person's name, a brand name, a government agency, a city, etc.
- *Sentiment assessment.* After this, the sentiment is determined: first for the text as a whole, and then, if possible, for individual entities. A hybrid approach is used here: a neural network model + reliance on emotional vocabulary.
- *Use of vocabulary.* The emotional vocabulary of the Uzbek language should not be kept "standalone", but should actually influence the result: it should strengthen or weaken the evaluation, assist in ambiguous cases, and take into account culturally specific expressions.
- *Working through a web interface.* For live use, a simple browser interface is required: a text input field, analysis results, and entity and sentiment highlighting directly in the text [17].
- *API support.* Those who want to integrate the system into their services need a programming interface: send a text request and receive JSON with the results [18].
- *Logging.* It's advisable to save a history of requests and responses so that you can analyze errors, look at typical examples, and further improve the system.

2.3. Non-functional Requirements: What's Important "Under the Hood"

Besides the functions themselves, there are several other aspects that are important for convenient real-world use of the system:

- *Speed*. Short text should be processed quickly enough so that the user doesn't forget what they just sent in the web interface.
- *Scalability*. If requests increase, the system should be easy to migrate to a more powerful server, connect a GPU, or run multiple instances simultaneously.
- *Flexibility and Development.* Models and vocabularies change over time. It's great if they can be updated and retrained without rewriting the entire system from scratch.
- *Installation and Deployment.* The system should be installed on a standard server infrastructure. Using containers (e.g., Docker) is a natural choice here, but in this text, we will focus only on the general principles.

- *Clear Interface.* Ideally, the system should be easy to use for people without deep machine learning expertise: analysts, customer service representatives, etc. The key is that the results are presented clearly and without unnecessary technical details.

All of this together defines the framework within which we design the system architecture. In the next section, we will describe how the modules interact and what components comprise the software system.

3. Software Architecture

When designing a system, it's important not only to select models but also to understand how all the components will work together. In our case, the system is built using a fairly transparent framework: there's a layer through which users and external applications access the system, an application server, and then a set of modules that process the text step by step.

3.1. General Operational Framework

In a typical situation, the user opens a web page, enters text in Uzbek, and launches the analysis. The external application operates similarly, except that instead of using a browser, it sends a request via a REST API. Both options ultimately lead to the same result: a request with text and a minimal set of parameters is sent to the application server.

The request is then passed on to the processing chain. First, the text is rectified (preprocessed), then named entities are identified, and finally, sentiment is calculated. A separate step is combining the results: the system needs to "bring together" the original text, the detected entities, and the sentiment scores. The result is generated in a convenient programmatic format (usually JSON) and returned either to the web interface or to an external system. In the web client, this result is transformed into a familiar form–highlighted text, tables, and short summaries.

The architecture can be roughly divided into three layers:

1. The client layer (the user's browser or external application).
2. The application server, which receives requests and manages processing.
3. The modular layer, which houses models and linguistic resources.

3.2. Application Server

The application server acts as a "central hub". It doesn't directly handle linguistics, but ensures the correct data flow. When the system starts, the server loads the necessary models, grants access to dictionaries, and reads the configuration. This eliminates the need for repeated initialization with each request.

When a request arrives, the server:

1. Checks that the data structure is correct (there is text, the required fields are not empty).
2. Passes the text to the processing modules.
3. Collects intermediate results into a single object.
4. Writes request and response information to the log if necessary.

A separate configuration layer is responsible for which specific models and lexicons are enabled, where logs are saved, and how resource paths are configured. This allows the system to be reconfigured without changing the core server logic.

3.3. Text Processing Modules

The main "intellectual" part of the system is concentrated in the text processing modules. Each of them solves its own subtask, and together they form a pipeline. The preprocessing module runs first. Its purpose is to slightly simplify the subsequent stages: remove technical symbols, remove unnecessary spaces, and carefully process links, mentions, and emoji. In some cases, it also

performs light normalization of frequently occurring spelling variations, especially when mixing Cyrillic and Latin.

Next in the chain is the NER module. It receives the already "cleaned" text and returns a list of found entities: the text fragment, its type, and its position in the source string. It's important to maintain a link to the original text, otherwise it will be difficult to correctly highlight entities in the interface and assign sentiment scores to them.

NER is followed by a sentiment analysis module. It uses a hybrid approach: a neural network model based on a transformer, and an Uzbek sentiment dictionary. The model provides a basic assessment, while the dictionary assists in questionable cases and enhances the signal when the text contains clearly evaluative vocabulary or culturally specific expressions. As a result, the module returns a sentiment label and, if necessary, numerical values that can be used for visualization.

The aggregation module completes the chain. It receives information from both the NER and the sentiment module and attempts to link them. For each entity, its surroundings in the text are analyzed, and, if possible, a local score is generated specifically for that entity. If the data is insufficient, the entity is at least marked as present in the text with a certain overall sentiment. The output is a structure that combines the source text, entities, global and local sentiment, and service information for display.

3.4. Web Interface and API

From a user perspective, the system appears quite simple. The web interface includes a text input field, a button to launch the analysis, and results display area. The text is highlighted: named entities are highlighted and can be associated with icons or color-coded sentiment labels. Brief summaries can also be displayed, such as a distribution by sentiment class or a list of entities with their type and a brief rating description.

For developers and integrators, the second method–via a REST API–is more important. In this mode, the external system sends a request with text and receives a JSON response with markup: a list of entities, sentiment ratings, and auxiliary data. This format allows the system to be integrated into an existing infrastructure, connecting it to review monitoring systems, analytics dashboards, or internal request processing services. Thus, the architecture of the complex remains fairly simple and transparent, but at the same time allows for development: models can be replaced, dictionaries updated, and the interface changed without disrupting the overall interaction scheme of the components.

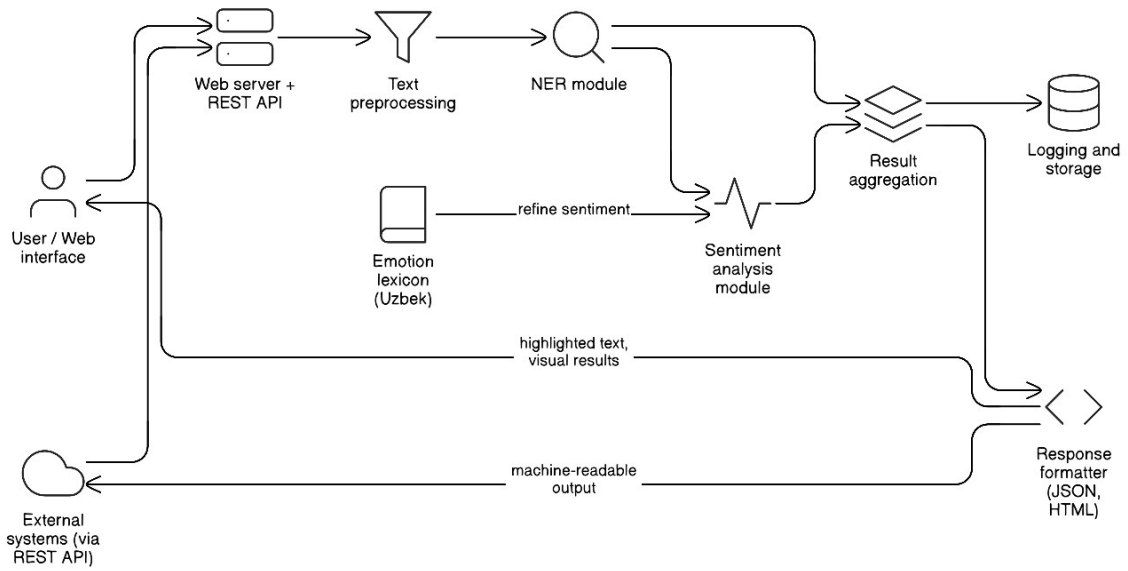

Fig. 1. Block diagram of the software system for hybrid sentiment analysis of Uzbek texts

Figure 1 illustrates the main components of the software suite and the data flow between them. Two types of clients are shown on the left: on one side, the user working through a web browser; on the other, external applications accessing the system via a REST API. Both types of clients transmit text and, if necessary, processing parameters to the application server. The application server acts as the entry point and coordinates further processing. It forwards the received text to sequentially executed modules. The first module performs preprocessing: it removes technical "noise", normalizes obvious spelling variations, and prepares the data for analysis, accounting for mixed graphics and the presence of emoji. The preprocessed text is then passed to the NER module, which extracts named entities (persons, organizations, geographic objects, etc.) and returns their types and positions in the source string. The text is then analyzed by a sentiment module based on a transformer model. The block diagram next to it shows the Uzbek emotional dictionary module, linked to the sentiment module by a side arrow. This dictionary is used to refine sentiment assessments and helps better process culturally specific emotional expressions. The outputs of the NER and sentiment analysis modules are combined in the results aggregation block. At this stage, where possible, links are established between specific entities and their corresponding sentiment values, and a unified representation of the analysis result is formed. The aggregated data is then fed to the response formatting module, which prepares either JSON (for the REST API) or structures suitable for display in the web interface. In the web client, the results are displayed as raw text, with entities and sentiment labels highlighted. At the same time, key information about requests and responses is recorded in the logging and storage module, facilitating system debugging and subsequent improvement.

4. Software Implementation Features

Unlike purely theoretical designs, the software suite had to be designed so that it could be run on a regular server and easily deployed again. In this section, we'll briefly describe the underlying technologies and how the code is organized.

4.1. Selecting the Technology Stack

We implemented the server-side component in Python, as this is the language for which the NER and sentiment analysis models had already been developed. A lightweight asynchronous framework (similar to FastAPI) was used as the web framework. This framework allows for the following:

1. implementing a standard web interface for the user;
2. providing a REST API for external applications;
3. operating in asynchronous mode and not blocking processing during multiple concurrent requests.

For working with neural network models, we used standard deep learning libraries (based on PyTorch) and a library of transformer models. This set is sufficient for loading ready-made models, retraining them if necessary, and performing inference on the CPU or GPU. On the frontend, a simple stack was used: HTML + CSS + a bit of JavaScript for sending requests to the server and dynamically displaying the results. No special heavy UI frameworks were required; the main goal was to make the interface as clear and uncluttered as possible.

4.2. Server-Side Organization

The server application runs once and, upon startup, loads all the resources needed for text processing: NER and sentiment models, an emotional dictionary, auxiliary tables, and configuration. This takes some time, but only happens once–all subsequent requests are served with these ready-made objects in memory.

The main elements of the server logic:

1) configuration module (paths to models, CPU/GPU operating mode, logging parameters);

2) module responsible for initializing models and dictionaries;

3) HTTP request handlers that accept text, call the necessary modules, and generate a response.

Each HTTP endpoint is as simple as possible: it validates the input data, calls the text processing function, and returns the result in JSON format. All the "smart" behavior is handled by a separate function that implements the pipeline: preprocessing → NER → sentiment → aggregation.

4.3. Representation of Models and Lexicon

The NER and sentiment analysis models are loaded as independent objects. They share a common interface: a list of strings is passed as input, and a list of structures with the required fields (label, probabilities, positions, etc.) is returned as output. This allows one model to be replaced with another without changing the rest of the code. The sentiment dictionary is stored as a file (JSON or CSV) and is loaded into memory at startup as a Python dictionary. For speed, normalized keys are used (converted to a single graph and base form, as far as possible). During the sentiment analysis stage, the lexicon is used as an additional source of information: scores for tokens and idioms are analyzed and factored into the final score. Some of the logic related to the lexicon is stored in a separate module. This allows for the subsequent integration of additional lexicons (e.g., for other domains or languages) without interfering with the neural network component.

4.4. Web Interface and REST API

The web interface is built on top of the same endpoints as the API. The only difference is how the results are displayed.

A typical workflow is as follows:

1. The user enters text into the form and clicks the "Analyze" button.
2. JavaScript on the page sends a request to the server (usually via fetch).
3. The server returns JSON with the text markup.
4. The client-side script converts this JSON to HTML:
5. breaks the text into fragments;
6. Wraps fragments with entities in with the required classes;
7. Adds sentiment markers (color, icon, short text comment).

External systems don't require any additional transformations: they work directly with the JSON response, which contains a field for the original text, a list of entities, and a block with sentiment information. This format is convenient for storing in a database or including in reports.

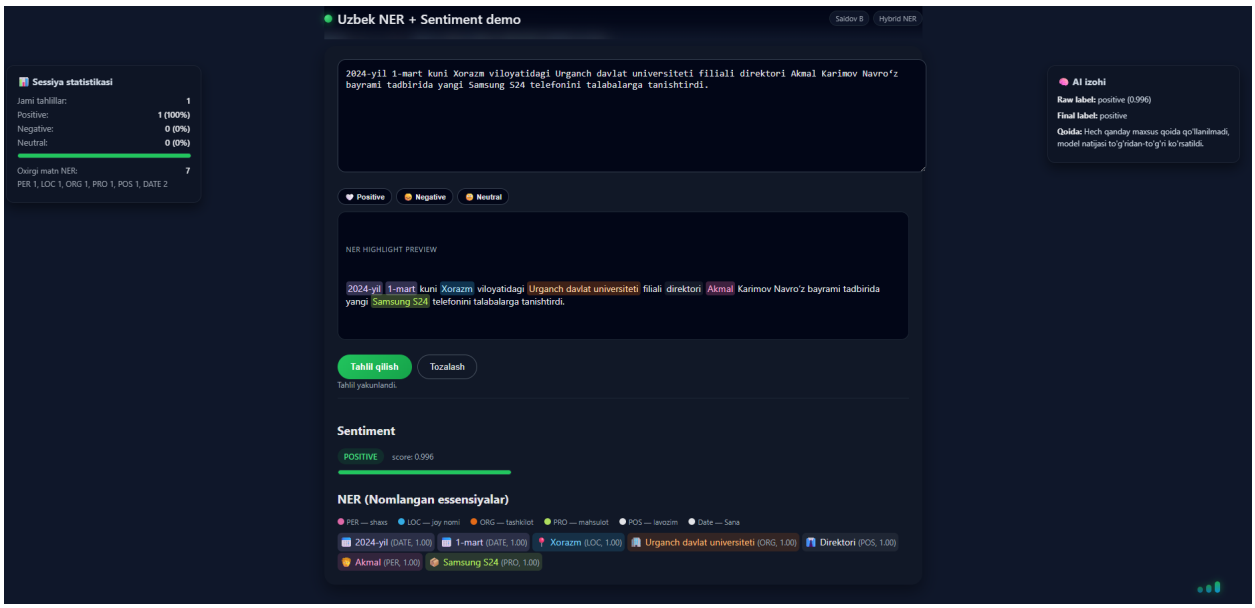

Fig. 2. Web interface of the hybrid sentiment and NER analysis system

The software suite's web interface is shown in Figure 2. The user enters text in the top field, after which the result is displayed in the central part of the window: highlighted named entities, the final sentiment label, and a brief summary of the analysis session. This interface is convenient for manual testing of the system, demonstrating its capabilities, and for analysts unfamiliar with the technical details of the implementation.

4.5. Logging and Testing

Special attention was paid to logging. For each request, we can (with debug mode enabled) save the following:

1. input text;
2. key processing steps (after preprocessing, after NER, after sentiment analysis);
3. final response sent to the user or external system.

This data helps us analyze cases where the system behaves differently than expected and gradually adjust models or rules. To test the robustness of the system, we used a small collection of test cases covering different types of text: reviews, short comments, messages with a large number of emojis, mixed graphics, etc. Some of these examples were used as regression tests: after code changes or model updates, we can quickly verify that the system's behavior in key cases has not deteriorated.

5. Experience And Results

In this section, we briefly describe how the software suite was tested, the data used, and the initial experiments' findings. We also provide several illustrative examples illustrating how the system performs on real texts.

5.1. Experimental Setup

For quality assessment, we used the same corpora used in previous studies on sentiment analysis and NER for the Uzbek language. Briefly, the dataset can be described as follows:

1) labeled texts with sentiment (positive, negative, neutral);
2) additional annotation of named entities (persons, organizations, locations, etc.);

3) examples containing informal writing, mixed Cyrillic and Latin script, emoji, and colloquial expressions.

The structure and main characteristics of the corpora used are summarized in Table I. It includes information on the data domain, the number of texts and tokens, as well as the distribution of sentiment classes and, where available, the annotation of named entities. This clearly demonstrates that the experiments were conducted on heterogeneous but comparable sample sizes.

Table I. Overview of the datasets used in the experiments

Dataset	Domain / Source	Texts	Tokens	Pos	Neg	Neu	NER entities
SA-main	Social media reviews	10 000	127 657	4 000	3 500	2 500	–
SA-emoji	Social media (emoji-rich)	3 000	42 123	...	...	...	–
NER-corpus	Mixed domains	5 000	76 559	–	–	–	18 328

Part of the corpus was set aside for model training, part for validation, and a separate part for final testing within the software suite.

At least two groups of models were considered for comparison:

1. A text-based sentiment analysis model that uses only the contextual representation of the text (without explicitly considering NER and the vocabulary).
2. A hybrid model integrated into the software suite: the same basic architecture, but supplemented with information about named entities and an emotional vocabulary.

The main metrics were accuracy and F1 score for sentiment classes. For NER, recall, precision, and F1 score were, as usual, assessed using the standard "token-level / span-level" scheme. However, in this article, we will limit ourselves to a discussion of the general trend without going into detail for all entity classes.

Table II separately presents statistics on the use of emoji and emoticons in the sentiment analysis corpus. It is clear that positive and negative emoji appear in a significant proportion of texts, and a layer of ironic and sarcastic markers is also present. This confirms that emoji are an important signal for determining sentiment and justifies their explicit modeling within the proposed framework.

Table II. Distribution of emoji and emoticons in the sentiment corpus

Category	Examples	#Occurrences	Share of texts (%)
Positive emoji	😊 😍 🤩	5 200	32.4
Negative emoji	😡 😢 😭	2 100	14.7
Irony / sarcasm emoji	😉 😁 😂	...	...
ASCII emoticons	:) :(:/	...	...

5.2. Quantitative Results

Overall, the experimental results were as expected: the hybrid approach integrated into the software suite demonstrates an advantage over the text-only model, especially for examples with pronounced evaluative vocabulary and named entities.

Several observations can be made:

1. In "simple" examples, where sentiment is obvious from just a few words, the difference between the models is small–both versions perform roughly equally.
2. In more complex texts, where the evaluation is spread across several sentences or addressed to different objects, the hybrid model produces more consistent responses.

The summary numerical results for the baseline text model and the hybrid model implemented in the software package are presented in Table III. The table includes the Accuracy and Macro-F1 values, as well as class-specific F1 metrics for positive, negative, and neutral texts.

Table III. Comparison of text-only and hybrid sentiment models

Model	Features used	Accuracy	Macro-F1	F1 (pos)	F1 (neg)	F1 (neu)
Text-only baseline	Transformer only	0.78	0.75	0.80	0.70	0.74
Hybrid (Ours)	Transformer + NER + lexicon	0.91	0.90	0.91	0.89	0.90

As Table III shows, the hybrid model consistently outperforms the baseline text model across all key metrics. The gain is particularly noticeable for the "negative" sentiment class, which is important for applied tasks like monitoring complaints and negative reviews. A slight improvement is also observed for neutral texts, indicating a more accurate discrimination of borderline cases.

The advantage is particularly noticeable in examples where the user directly mentions organizations, services, cities, etc., and the evaluation is formulated not directly, but through indirect formulations and comparisons. According to the average performance on the test sample, the hybrid model yields a higher F1 value for the "negative" class and slightly improves performance for "neutral" texts.

This is important, as most practical monitoring tasks typically involve negative and ambiguous reviews. It is important to emphasize that the goal of these experiments was not so much to achieve record metric values, but rather to verify that the architecture of the complex, in principle, allows the use of combined information (context + NER + lexicon) and that this provides an advantage compared to the basic text model.

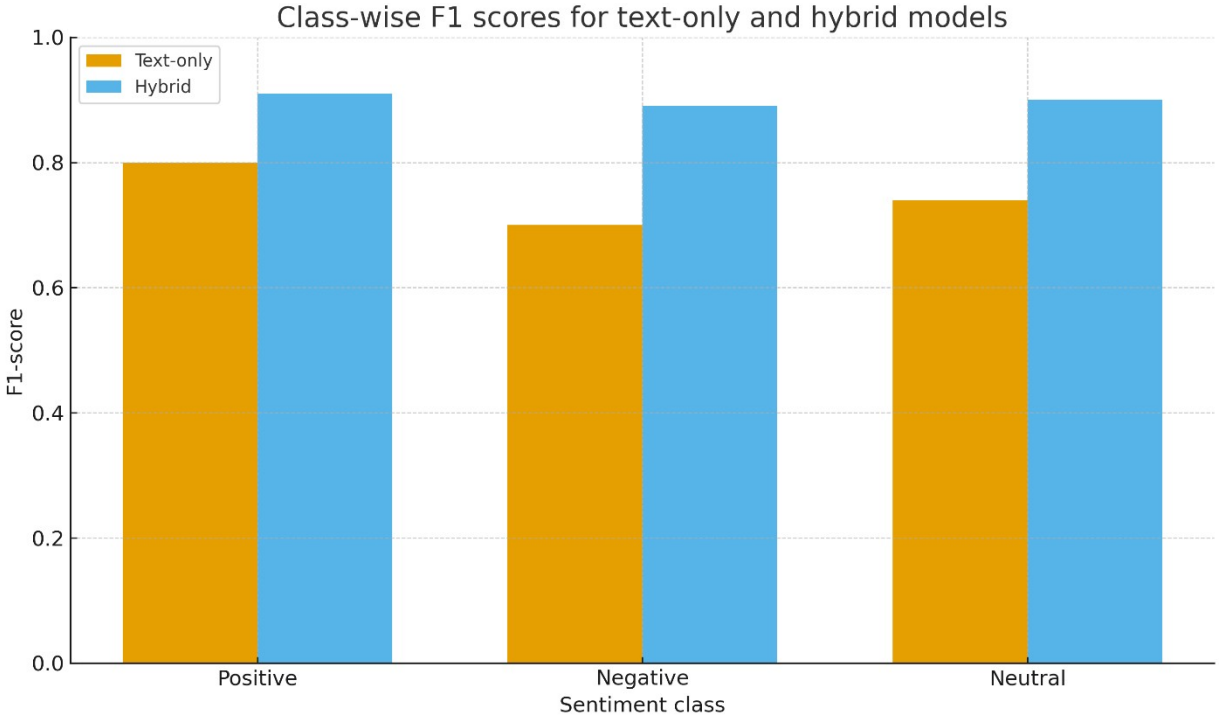

Fig. 3. Class-wise F1 scores for text-only and hybrid sentiment models

Class-specific F1 values for the text and hybrid models are shown in Figure 3. Paired columns are shown for each sentiment class (positive, negative, neutral), allowing for a visual assessment of the hybrid approach's gains over the baseline model. The increase in F1 for the negative class is particularly noticeable, consistent with the numerical data in Tables III and IV and highlighting the practical value of considering NER and lexicon when working with complaints and negative reviews.

The final hybrid NER + BERT model was evaluated on the held-out test split containing 40 000 texts. On this set, the model achieved an overall accuracy of 0.910, with macro precision of 0.905, macro recall of 0.896 and a macro F1-score of 0.900. Per-class F1-scores were 0.91 for the POSITIVE class, 0.89 for the NEGATIVE class and 0.90 for the NEUTRAL class. These results

are consistent with the relative improvements shown in Tables III and IV: the hybrid architecture outperforms the text-only baseline and the ablation variants not only in terms of aggregate metrics, but also across all sentiment classes. In particular, the gain for the NEGATIVE class is important from a practical point of view, since this class is most closely associated with user complaints and critical feedback.

5.3. System Operation Examples

To better understand how this all looks in real life, we'll provide a few typical examples. These examples don't show formal metrics, but rather what the user sees in the web interface.

Example 1. Positive review of a service

Text (hypothetical example):

"*Namangan branch office is very positive, and I'm very happy with my service* 😊"

The system identifies:

1) entity: "Namangan branch office" → type "location / organization branch";
2) overall sentiment: positive;
3) local sentiment towards the branch: positive.

In the interface, the user sees that the text as a whole is rated positive, and the mentioned branch is highlighted as an object with a positive rating. The presence of emoji further enhances the model's confidence.

Example 2. Mixed Rating

Text (hypothetical example):

"*Shahar o'zi yoqadi, lekin avtobuslar grafigi bilan muammo juda ko'p*."

In this case, the system detects that:

1) "Shahar" (the specific name, if specified) receives a positive rating.
2) "avtobuslar grafigi" and the related service are in a negative context.
3) The overall conclusion is a mixed, but rather negative, tone to the text, since the main dissatisfaction is related specifically to transportation.

This is important for the user: it is clear that the message is not "bad about the city in general", but rather about a specific problem related to one area.

Example 3. Informal message with emoji

Text (hypothetical example):

"*Bu bank bilan ishlashdan charchadim* 😡*, navbatlar tugamaydi, online ilova ham sekin*".

The system identifies:

1) entity "Bu bank" → type "organization (bank)";
2) overall tone: clearly negative;
3) emotional vocabulary and emojis reinforce the "negative" signal;
4) additionally, it can be noted that the complaint is related to the quality of service and the operation of the online application.

In the visualization, the bank is highlighted as the main object of the negative rating, and the text around it is displayed with the corresponding color coding.

To assess the contribution of the individual components of the hybrid approach, an ablation analysis was conducted, the results of which are presented in Table IV. Four model variants were considered: a text-only model, a model considering only NER, a model using only the lexicon, and a full hybrid version combining both information sources.

Table IV. Ablation study of sentiment models

Модель	NER used	Lexicon used	Accuracy	Macro-F1
Baseline (text-only)	–	–	0.78	0.75
Baseline + NER	✓	–	0.82	0.80
Baseline + Lexicon	–	✓	0.81	0.79
Hybrid (NER + Lexicon, ours)	✓	✓	0.91	0.90

Table IV shows that adding either component (NER or lexicon) improves performance compared to the baseline model, but the best results are achieved when using them together. This confirms that named entity information and the emotional vocabulary complement each other and should be considered as a unified mechanism for enhancing the model.

For ease of comparison between the models, individual illustrative examples are summarized in Table V. It includes shortened text fragments, reference sentiment labels, and the responses of the text-only and hybrid models.

Table V. Examples of model behaviour on real texts

Example	**Short text (fragment)**	**Gold label**	**Text-only model**	**Hybrid model**
1	“Namangan branch office service was great 😊”	Positive	Positive	Positive
2	“Shahar o‘zi yoqadi, lekin avtobuslar grafigi bilan…”	Negative	Neutral	Negative
3	“Bu bank bilan ishlashdan charchadim 😡, navbat tugamaydi”	Negative	Negative	Negative

Table V clearly demonstrates that the hybrid model performs better in complex and mixed cases. In the second example, the baseline model tends to underestimate the degree of negativity and produces a neutral verdict, while the hybrid version correctly records the negative rating, taking into account the structure of the statement and the presence of a specific object of criticism.

5.4. Runtime performance and resource usage

In addition to accuracy, the hybrid system's time-to-response performance was measured. The evaluation was performed on the same test subset of 40 000 texts used for the quantitative analysis. The experiments were conducted on a machine equipped with an NVIDIA RTX 4070 GPU (8 GB), an Intel Core i7-12700 processor, and 32 GB of RAM.

With this hardware configuration, the average end-to-end processing time for a single text–from the moment the request arrives at the application server to the generation of the final JSON response–was 0.18 seconds. This time includes all stages of the pipeline: preprocessing, named entity extraction (NER), sentiment analysis and result aggregation [19]. The corresponding console output of the evaluation script is shown in Figure 4.

These results demonstrate that the current implementation is fast enough for interactive work in the web interface and for processing moderate user message flows in near real time. Since all models and resources are loaded once at system startup and reused for subsequent requests, the main influence on processing time is the inference of the neural network model and the length of the input text, and the overhead of initialization is practically insignificant.

```
(venv) C:\projects\uzbek-hybrid-ner-sentiment> python evaluate_hybrid_model.py -
split test
[INFO] Loading test set (40 000 texts)...
[INFO] Restoring model from: models/hybrid_ner_sentiment/best_checkpoint.pt
[INFO] Running evaluation...

Evaluation finished.
----------------------------------------
Overall results (Hybrid NER + BERT model)
----------------------------------------
Test Accuracy:      0.910
Macro Precision:    0.905
Macro Recall:       0.896
Macro F1-score:     0.900
----------------------------------------
Per-class F1:
  POSITIVE : 0.91
  NEGATIVE : 0.89
  NEUTRAL  : 0.90
----------------------------------------
Average inference time per text: 0.18 s
Device: NVIDIA RTX 4070 (8 GB), CPU: Intel Core i7-12700, RAM: 32 GB
```

Fig. 4. Console output of the hybrid NER + BERT sentiment model evaluation on the 40 000-text test set

5.6. Error analysis and typical failure cases

To better understand the limitations of the proposed system, a brief analysis was conducted of cases where the hybrid model still produced incorrect or unstable predictions. The analysis was performed on the same test set used for quantitative quality assessment.

Several recurring error types were identified:

1. **Sarcasm and irony.** In texts where the author intentionally uses positive language to express a negative attitude (and vice versa), both the base and hybrid models make mistakes. In such situations, emotional vocabulary often provides a misleading signal, and the local context is insufficient to correctly reconstruct the true meaning of the statement.
2. **Long messages with multiple topics.** If several different storylines are discussed simultaneously in a single message (for example, transportation, prices, and employee attitudes within a single text), the system tends to focus on the dominant sentiment and underestimate the nuances in less noticeable parts of the message. As a result, the prediction may be partially correct at the document level, but from the user's perspective, the overall picture remains incomplete [20].
3. **Domain switching and rare entities.** In some cases, the model struggles with industry-specific terminology or rare named entities that are poorly represented in the training data. For example, highly specialized company names or new services may be correctly identified by the NER module, but the sentiment model has almost no a priori information about their typical contexts.
4. **Highly noisy or mixed (code-switched) input.** Messages with extensive code-switching between Uzbek, Russian, and English, extensive use of slang, and non-standard spellings remain challenging to process. Although preprocessing and normalization steps partially reduce variability, in extreme cases of noise, the model's predictions become unstable [21].

These observations do not call into question the overall effectiveness of the hybrid approach, but highlight areas for further development of the system. In particular, expanding the training data with examples of sarcastic usage, improving coverage of industry-specific vocabulary, and more robust handling of cases of intensive language switching appear promising.

6. Conclusion

This paper describes a software suite that combines several previously disparate components–sentiment analysis models, a named entity extraction module, and an Uzbek language sentiment dictionary–into a single applied solution. The system is designed for real-world texts:

short messages, reviews, and social media comments, which mix various scripts, colloquial vocabulary, and emoji [22].

It is shown that the hybrid approach implemented in the suite offers advantages over a purely text-based model: taking into account named entities and lexicon allows for more stable processing of texts in which the evaluation is addressed to specific organizations, services, or geographic objects. At the same time, the architecture of the suite remains relatively simple: individual modules (preprocessing, NER, sentiment, and aggregation) are linked via a transparent pipeline, and the system can be accessed via both a web interface and a REST API. From a practical standpoint, the developed suite can be considered the basis for applied systems for monitoring reviews and requests in the Uzbek language [23]. It is already suitable for experiments and pilot implementations, and if necessary, it can be adapted to specific data domains by updating models and dictionaries without a major redesign of the architecture.

However, this work does not resolve all open questions. Further expansion and refinement of the corpora, improved processing of informal writing, and more precise matching of sentiment to entities in complex multi-object texts are required. A separate area of development is the transfer of the underlying principles to other Turkic languages, which face similar challenges with resources and writing practices.

Nevertheless, even at this stage, the developed software suite demonstrates that the combination of modern deep learning models with targeted linguistic resources yields practical results and can serve as a working tool for analyzing Uzbek texts in applied information systems.

References

1. Kuriyozov E., Salaev U., Matlatipov S., Matlatipov G. Text classification dataset and analysis for Uzbek language. *arXiv preprint* arXiv:2302.14494, 2023. DOI: 10.5281/zenodo.7677431.
2. Matlatipov S., Rahimboeva H., Rajabov J., Kuriyozov E. Uzbek Sentiment Analysis Based on Local Restaurant Reviews. In: *Proceedings of the ALTNLP Workshop*. 2022. DOI: 10.48550/arXiv.2205.15930.
3. Matlatipov S., Rajabov J., Kuriyozov E., Aripov M. UzABSA: Aspect-Based Sentiment Analysis for the Uzbek Language. In: *Proceedings of SIGUL 2024 – 3rd Workshop on Speech and Language Technologies for Low-Resource Languages*. 2024.
4. Kuriyozov E., Vilares D., Gómez-Rodríguez C. BERTbek: A Pretrained Language Model for Uzbek. In: *Proceedings of SIGUL 2024*. 2024. P. 33–44.
5. Tan K. L., Lee C. P., Lim K. M. A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research. *Applied Sciences*, 2023, vol. 13, no. 7, 4550. DOI: 10.3390/app13074550.
6. Nip J. Y. M. Social Media Sentiment Analysis. *Encyclopedia*, 2024, vol. 4, no. 4, 104. DOI: 10.3390/encyclopedia4040104.
7. Aftab F., Bazai S. U., et al. A Comprehensive Survey on Sentiment Analysis Techniques. *International Journal of Technology*, 2023, vol. 14, no. 6. DOI: 10.14716/ijtech.v14i6.6632.
8. Aliyu M. A., Ibrahim F., Musa A., et al. A Systematic Review of Sentiment Analysis in Low-Resource Languages. *IEEE Access*, 2024, vol. 12. DOI: 10.1109/ACCESS.2024.3398635.
9. Mengliev D., Abdurakhmonova N., Shirinova R., Suyunova M. Developing Named Entity Recognition Algorithms for Uzbek: Dataset Insights and Implementation. *Data in Brief*, 2024, vol. 54, 110413. DOI: 10.1016/j.dib.2024.110413.
10. Saidov B. R., Barakhnin V. B., Rixsibayev U. T., Sharipov E. J. Methods of Automatic Selection of Named Entities (NER) in Uzbek Language for Text Tone Analysis. In: *Proceedings of the International Conference on Computational Linguistics and Intelligent Systems*. 2025.

11. Saidov B. R., et al. Creating an Emotional Dictionary for the Uzbek Language: Taking Cultural–Linguistic Specifics into Account. *Journal of Language Resources and Evaluation*, 2025, vol. 59, no. 2, pp. 245–268.
12. Saidov B. R., et al. Sentiment Analysis of Uzbek Texts Using NER: A Comparative Study of SVM, LSTM, and BERT Models. *Vestnik SibGUTI*, 2025, vol. 19, no. 4, (in press).
13. Atlas L. G., et al. A modernized approach to sentiment analysis of product reviews using BiGRU and RNN based LSTM deep learning models. *Scientific Reports*, 2025, vol. 15, 6642. DOI: 10.1038/s41598-025-01104-0.
14. Chhetri G., Dutta A., Das S. CognitiveSky: Scalable Sentiment and Narrative Analysis for Decentralized Social Media. *arXiv preprint* arXiv:2509.11444, 2025. DOI: 10.48550/arXiv.2509.11444.
15. Islam T., et al. Lexicon and Deep Learning-Based Approaches in Sentiment Analysis on Short Texts. *Journal of Computer and Communications*, 2024, vol. 12, no. 1, pp. 11–34. DOI: 10.4236/jcc.2024.121002.
16. Tang X., et al. EMFSA: Emoji-based multifeature fusion sentiment analysis. *PLOS ONE*, 2024, vol. 19, no. 9, e0311415. DOI: 10.1371/journal.pone.0311415.
17. Sufi F. K. AI approach on identifying change in public sentiment for major events: Dubai Expo 2020. *Journal of Engineering Research*, 2024. DOI: 10.1016/j.jer.2024.07.010.
18. Hash Block. Real-Time Sentiment Analysis App with NestJS Backend and Streamlit Frontend. *Medium*, 11 July 2025. Available at: https://medium.com/...
19. Seow W. L., Chaturvedi I., Hogarth A., Mao R., Cambria E. A review of named entity recognition: from learning methods to modelling paradigms and tasks. *Artificial Intelligence Review*, 2025, vol. 58, 315. DOI: 10.1007/s10462-025-11321-8.
20. Veitsman Y., Hartmann M. Recent Advancements and Challenges of Turkic Central Asian Language Processing. In: *Proceedings of the First Workshop on Language Models for Low-Resource Languages (LoResLM)*, 2025. DOI: 10.48550/arXiv.2407.05006.
21. Saidov B. R., Barakhnin V. B., Madirimov Sh., Ibragimov U., Meylikulov Sh., Normamatov S., Bahodirova F., Matnazarov J., Fayzullaeva Z. Dual-Source Synthetic Uzbek Corpora for Sentiment Analysis and NER with Controlled Emoji Signals. *Data*, 2026, vol. 11, no. 2, 28. DOI: 10.3390/data11020028.
22. Saidov B. R., Barakhnin V. B., et al. Methods of Automatic Selection of Named Entities (NER) in Uzbek Language for Text Tone Analysis. In: Proceedings of the IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM-2025). Altai, Russian Federation, 2025. DOI: 10.1109/EDM65517.2025.11096748.
23. Saidov B. R., et al. Integration of a Sentiment Dictionary and NER System into a Tool for Analyzing Public Opinion in the Uzbek Language. In: Proceedings of the IEEE 2025 International Conference AL-KHWARIZMI (APEIE-2025). Urgench, Uzbekistan, 2025. DOI: 10.1109/APEIE66761.2025.11289303.

Bobur R. Saidov

PhD student, Novosibirsk State University (630090, Russia, Novosibirsk, Pirogova, 1), e-mail: b.saidov@g.nsu.ru. ORCID: 0009-0000-5540-2013.

Vladimir B. Barakhnin

Doctor of Technical Sciences, Associate Professor, Novosibirsk State University (630090, Russia, Novosibirsk, Pirogova, 1). Federal Research Center for Information and Computational Technologies (630090, Russia, Novosibirsk, Lavrentieva avenue, 6), e-mail: v.barakhnin@g.nsu.ru. ORCID: 0000-0003-3299-0507.

Программный комплекс для гибридного анализа тональности узбекских текстов с использованием именованных сущностей

Б. Р. Саидов[1], В. Б. Барахнин[2,3]

[1]Новосибирский национальный исследовательский государственный университет
[2]Федеральный исследовательский центр информационных и вычислительных технологий
[3]Сибирский государственный университет телекоммуникаций и информатики (СибГУТИ)

Аннотация: В данной статье представлен программный комплекс для автоматического анализа тональности узбекских текстов. Система основана на гибридном подходе, сочетающем модель трансформатора, модуль извлечения именованных сущностей (NER) и специально составленный словарь тональности узбекского языка. Актуальность данной разработки обусловлена растущим объемом неформальных текстов в социальных сетях и отсутствием готовых инструментов для их обработки. Комплекс реализует полный цикл обработки: очистку и нормализацию текста, извлечение сущностей, определение тональности и ключевых слов, а также визуализацию результатов во встроенном веб-интерфейсе. Модели и словарь адаптированы к агглютинативным и орфографическим особенностям узбекского языка, что повышает устойчивость к разговорной и смешанной письменности. Кратко описаны архитектура комплекса, основные программные модули и их взаимодействие, а также принцип работы прикладного интерфейса (REST API). Приведены примеры использования системы для анализа отзывов и сообщений пользователей, подтверждающие её пригодность для решения прикладных задач мониторинга общественного мнения. По результатам первоначальных экспериментов достигнуто существенное улучшение качества по сравнению с базовыми моделями текста без учета НЭР и лексики.

Исследование выполнено в рамках государственного задания Министерства науки и высшего образования Российской Федерации для Федерального исследовательского центра информационных и вычислительных технологий.

Ключевые слова: анализ тональности; узбекский язык; извлечение именованных сущностей; эмоциональный словарь; гибридная модель; программный комплекс.

Для цитирования: Саидов Б. Р., Барахнин В. Б. Программный комплекс для гибридного анализа тональности узбекских текстов с использованием именованных сущностей // Вестник СибГУТИ. 2026. Т. 20, № 1. С. 23–38. https://doi.org/10.55648/1998-6920-2026-20-1-23-38.

Статья поступила в редакцию 04.12.2025;
переработанный вариант – 15.12.2025;
принята к публикации 05.02.2026.

Саидов Бобур Рашидович
аспирант НГУ (630090, Новосибирск, ул. Пирогова, 1), e-mail: b.saidov@g.nsu.ru, ORCID: 0009-0000-5540-2013.

Барахнин Владимир Борисович
доктор технических наук, доцент НГУ (630090, Новосибирск, ул. Пирогова, 1); Федеральный исследовательский центр информационных и вычислительных технологий (630090, Россия, Новосибирск, пр. Лаврентьева, 6), e-mail: v.barakhnin@g.nsu.ru. ORCID: 0000-0003-3299-0507.