Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

DOI: 10.37943/21CPQX5616

Aigerim Mussina
PhD student of Computer Science, Department of Computer Science
mussina.aigerim95@gmail.com, orcid.org/0000-0002-7043-0810
Al-Farabi Kazakh National University, Kazakhstan

Didar Yedilkhan
PhD, Head of the Scientific and Innovation Center "Smart City"
d.yedilkhan@astanait.edu.kz, orcid.org/0000-0002-6343-5277
Astana IT University, Kazakhstan

Yermek Alimzhanov
Master of Mathematics, Director of Digital Institute of Lifelong Education
ermek.alimzhanov@astanait.edu.kz, orcid.org/0000-0002-8758-2220
Astana IT University, Kazakhstan

Aliya Nugumanova
PhD, Head of the Scientific and Innovation Center "Big Data and Blockchain Technologies"
a.nugumanova@astanait.edu.kz, orcid.org/0000-0001-5522-4421
Astana IT University, Kazakhstan

Sanzhar Aubakirov
PhD, Department of Computer Science
aubakirov.sanzhar@gmail.com, orcid.org/0000-0002-8416-527X
Al-Farabi Kazakh National University, Kazakhstan

Aigerim Mansurova
Master of Technical Sciences
222215@astanait.edu.kz, orcid.org/0009-0003-1978-9574
Astana IT University, Kazakhstan

USING MLOPS FOR DEPLOYMENT OF OPINION MINING MODEL AS A SERVICE FOR SMART CITY APPLICATIONS

Abstract: This paper presents the MLOps strategy, which adapts the automation principles of DevOps to the deployment and lifecycle management of artificial intelligence (AI) models. By leveraging high-performance automation, MLOps ensures seamless AI development and operations integration, enabling efficient and reliable model deployment. The study demonstrates this approach by implementing the Astana Opinion Mining macro-service customized for sentiment analysis. This macro-service evaluates public opinions based on a criteria taxonomy for assessing the urban environment's sustainable development. As a smart city application, the system facilitates the collection and analysis of citizen feedback to assess the performance of city services and inform urban planning decisions. Technologically, the MLOps strategy employs containers and microservices to construct robust data and process pipelines. Four core pipelines were developed in this research: data collection, feature engineering, experimentation, deployment, and maintenance. The data collection pipeline is achieved through automated crawling from diverse sources such as social media and other internet platforms. The feature engineering pipeline ensures data preprocessing by removing noise, identifying message languages, categorizing topics, and preparing data for further analysis. The experimentation pipeline incorporates services for data labeling, model training, and performance

Received: 07.12.2024 Accepted: 24.03.2025 Published: 30.03.2025

DOI: 10.37943/21CPQX5616

evaluation customized to sentiment analysis tasks. Finally, the deployment pipeline and maintenance pipeline deliver trained models to end-users, ensuring their continual improvement and adaptation. Using this MLOps framework, four models of sentiment analysis were tested in Russian: «Blanchefort,» «Sismetanin,» «MonoHime,» and «Dostoevsky.» The «Blanchefort» showed an accuracy of 71,43%. The resulting MLOps framework is fault-tolerant, scalable, and enables real-time urban environment assessments. By automating workflows, the architecture enhances operational efficiency, offering practical applications for smart city initiatives and sustainable urban development, contributing to better decision-making.

Keywords: Sentiment Analysis; Smart City; Urban Environment; Opinion Mining; Microservice Architecture

Introduction

The current decade is characterized by a steady increase in the number of software products that use advanced achievements of artificial intelligence. More and more solutions based on machine learning are moving from the category of research projects to the category of industrial applications. Accordingly, more work is being devoted to accelerating and standardizing the development of such applications. At the same time, the changeable and experimental nature of machine learning often contradicts the requirements of reliability, explainability, reproducibility, and scalability imposed on the quality of software products. It thereby causes a gap between the process of developing machine learning models and the process of their operation [1]. MLOps, as a methodology for optimizing the life cycle of machine learning, is a paradigm designed to bridge this gap [2], [3]. Its goal is to manage the quality of production and operation of machine learning models [4], [5].

In this paper, the MLOps is employed methodology to design the Astana Opinion Mining macro service, which collects, processes, and analyzes feedback from Astana residents published on social media platforms regarding the performance of municipal utilities, transportation, social, and other services. The primary objective of this service is to support decision-making at the operational level, providing city authorities with an understanding of citizens' specific needs while highlighting any shortcomings in the city's services. Additionally, this service aims to contribute to the overall assessment of the city's sustainable development at the strategic level according to the SULPiTER methodology to calculate the sustainability index of the urban environment of Astana in the concept of smart cities [6], forming part of a digital framework for urban planning and management alongside other smart city applications.

Our work is motivated by research on the bottom-up concept of smart cities development, which emphasizes the importance of open innovation processes [7], [8] and crowdsourcing [9], [10] as key mechanisms for managing change. Active and passive crowdsourcing are considered promising tools for both evaluating and supporting management and planning processes in the context of smart cities development [11], [12]. Passive crowdsourcing, which involves collecting big data from social networks, covers a much larger audience than active crowdsourcing, which is represented by specialized mobile applications, electronic voting systems, and other means.

According to [8], passive crowdsourcing has significant potential due to the abundance of unused and unaffected open data on the Internet. However, in terms of data quality, passive crowdsourcing is significantly inferior to active crowdsourcing since big data in social networks are often heterogeneous, noisy, unrepresentative, or unreliable [13], [14]. Accordingly, developing an opinion extraction service that can function flawlessly in a real work environment, processing heterogeneous, noisy, and contradictory data streams from social networks, goes far beyond the standard machine learning process launched in a local academic system

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

[15]. This study of leveraging MLOps methodology and passive crowdsourcing hypothesizes that a modular NLP-driven opinion mining system can efficiently process and analyze noisy social media data to provide actionable insights for smart city decision-making at operational and strategic levels.

The contribution of our work lies in the fact that we use the MLOps methodology; we implement the modular architecture of the Astana Opinion Mining macro-service, customized for various tasks and infrastructure facilities, based on the taxonomy of criteria for sustainable development of the urban environment. For each criterion, a list of keywords in two languages (Kazakh and Russian) is automatically generated based on Topic Modeling, according to which reviews are searched, and a list of sites and social media is set where these reviews can be found. The feedback obtained by the crawler is fed into the NLP pipeline, where it undergoes processing, validation, relevance assessment, and classification based on a defined criteria taxonomy. The resulting texts are further analyzed using the trained Opinion Mining model to extract aspect-oriented polarities. A diagram of the data flows for the proposed macro service is presented in Figure 1.

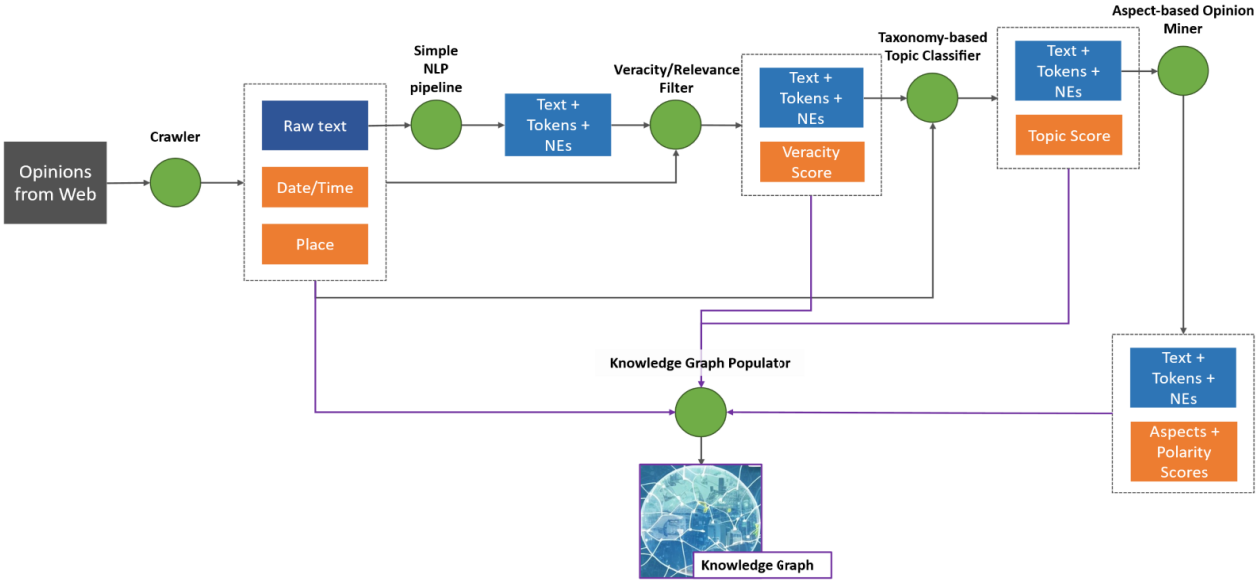


Figure 1. Data Flow Diagram for a macro service for collecting, processing and analyzing feedback from citizens of Astana Opinion Mining.

The macro service can be accessed through two methods: 1) through the web interface as a tool for collecting and measuring public opinion in the context of customizable tasks and objects; 2) through the API as a module within smart city applications to support decision-making at the operational and strategic level.

Literature Review

MLOps is an emerging field of research, and existing works can be broadly categorized into three classes: 1) methodological research on the ideology, concepts, trends, and challenges of MLOps; 2) instrumental research that focuses on MLOps development platforms and technologies; 3) applied research that describes specific cases of MLOps application. This classification is consistent with the definition of MLOps given in [16], according to which MLOps includes several aspects, such as (a) conceptual attitudes, (b) development culture, (c) best practices.

DOI: 10.37943/21CPQX5616

a. Methodological research

The work of [4] is worth noting, as it significantly contributes to the systematization of MLOps research. The authors of this work developed an MLOps taxonomy to better understand the methods and prospects of this emerging field of knowledge. Based on the most important conclusions obtained from the taxonomy, they are trying to formulate a new standard for machine learning projects. While data preparation, training, and testing have traditionally been considered the key elements of pipelines in machine learning projects, the new standard emphasizes the importance of continuous monitoring, sustainability assessment, and explicable artificial intelligence. The proposed standard also attaches special importance to key performance indicators (KPIs), which should correspond to the metrics used by data specialists to assess the quality of the models being developed.

The work [17] contributes to forming MLOps standards by offering a three-level meta-model that defines the primary artifacts for designing MLOps-based applications. At the upper level, the operational context is defined, describing the conditions under which the transfer of models can be carried out. For example, a task can be reused in a new context if the task structures are equivalent, even if the domains have different feature spaces. This level also contains detailed information about the models involved in the work, and the models themselves, in turn, are determined by the optimal tuning of their hyperparameters. At the middle level, operational policies regarding security, privacy, performance, and model monitoring are set. Finally, instances of technological components of machine learning are defined at the lower level.

In [18], the MLOps methodology is considered from the point of view of data quality. The authors point out that MLOps problems are often associated with data management problems, given the strong dependence of machine learning models' quality on the source data quality. They are trying to develop a fundamentally new MLOps methodology based on a joint understanding of data quality and subsequent (downstream) machine learning processes. At the same time, as the authors note, their methodology requires not only novelty but also practical value for a much larger number of machine learning scenarios and models than were considered in their previous works (see, for example, [19], [20], [21]).

In [22], a pipeline based on traditional DevOps practices is proposed, which includes the stages of developing machine learning models. Compared to traditional DevOps, the pipeline has a new validation module that supports two additional cycles: optimization and debugging. Suppose the module determines that the machine learning model is unsuitable for the available data or algorithms. In that case, it returns to the planning stage to optimize the model. And only after optimization and debugging the machine learning code is integrated into the overall system code.

In [23], approaches to the organization of the life cycle of a machine learning project are considered from the standpoint of technical, organizational, behavioral, and temporal aspects. The authors consider MLOps projects as sociotechnical systems and distinguish four levels of training in the life cycle of these projects: 1) machine learning; 2) training users to work with the pipeline; 3) training the project team and/or organization in continuous development and operation; 4) training the project team and/ or organization to improve and re-engineer design solutions to extract new ideas or values. Thus, another step is being taken toward explicable, responsible, and human-centric artificial intelligence. [24] also discusses the problems of understanding MLOps in the context of responsible artificial intelligence and offers a guide to action for implementing responsible MLOps, consisting of 5 steps.

In [25], the analysis of the literature on the organization of machine learning projects is carried out in the form of a search for answers to 3 research questions: 1) what are the problems in the development of machine learning projects; 2) why exactly MLOps is a concept designed to solve these problems, what is its difference from DevOps; 3) how to apply the key

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

principles of DevOps to MLOps. The authors identify two main problems in designing machine learning systems: at the management level and the dependency level. At the management level, the problem is caused by the heterogeneity of machine learning system components, which may require different environments, configurations, or tools. The authors note that managing the entire life cycle of machine learning, when each part is performed in its environment, is a difficult task. At the dependency level, one of the biggest problems is data, which is unstable. As in [18], it is emphasized here that the data used for training and testing are crucial in machine learning projects since an error occurring in the data can lead to a completely different system behavior. In order to use reliable data, the quality of which is guaranteed, the authors pay special attention to the problems of implementing data versioning.

b. Instrumental research

One of the most detailed reviews of development tools for MLOps projects is presented in [26]. The authors proceed from the position that currently, there is not a single open-source tool that can create a fully automated MLOps pipeline, and therefore, consider all technologies and tools separately for each part of the pipeline. The paper also deeply analyzes the roles of pipeline participants, and it is noted that if earlier the role of a data specialist was limited to experiments and modeling, now close cooperation with development or operation engineers is necessary for the successful deployment of a machine learning model. After discussing the requirements for the pipeline implementation, the authors tested 26 open-source tools and form a ranked list of tools based on the developed performance indicators.

In [27], the authors compared eight commercially available MLOps frameworks in terms of functionality: data versioning, hyperparameter configuration, model versioning and experiments, pipeline versioning, availability of continuous development and delivery, model deployment, and performance monitoring. Frameworks were also compared according to such an important indicator as supported programming languages and libraries. The authors found that most frameworks had limitations in providing developers with automated MLOps dashboards based on the user interface. Accordingly, this limitation should encourage researchers to expand the horizons of knowledge in the direction of integrated MLOps automation.

In [28], a comprehensive literature review is conducted to identify and analyze tools supporting the creation of MLOps pipelines. The study includes 13 MLOps development tools evaluated by three groups of characteristics: 1) main features (including openness, scalability, elasticity, support for continuous integration and delivery, usability, and API support); 2) data management functions; 3) model management functions. The analysis results show that most MLOps tools support the same functions but use different approaches that can give different advantages depending on the user's requirements. As a future work, the authors plan to develop methods that facilitate the integration of various MLOps tools.

There are works in which the authors propose their platforms for implementing MLOps concepts. In [29], a framework is proposed that helps to implement MLOps best practices on an industrial scale. The framework is assembled from modules that can be recomposed to adapt to different use cases. All datasets and machine learning artifacts (for example, models) are carefully monitored and versioned within the framework. The machine learning experiment process is standardized, and two cloud providers are presented to scale the learning and forecasting process in an industrial environment.

In [30], the Edge MLOps framework was developed, which supports boundary computing and provides continuous training on the model, its deployment, delivery, and monitoring. The framework successfully passed the test tests on predicting air quality, automatically retraining, integrating, and deploying models when their performance fell below a certain threshold. All trained and retrained machine learning models have been serialized in the Open Neural Network Exchange (ONNX) format, which ensures model compatibility when exporting

and deploying to production environments. In [31], the Tiny-MLOps framework is proposed, designed to specialize standard practices for managing machine learning models, including their deployment on distant peripheral devices. The developers have adapted all the stages of model orchestration to the limited resources available on typical IoT devices.

In [32], the Pangea tool is presented, designed to automatically create suitable runtime environments for deploying analytical pipelines in the mining industry. These pipelines are divided into different stages to perform each in the most suitable environment (peripheral, foggy, cloud, or local), optimizing the use of hardware and software resources. Pangea allows one to create the necessary infrastructure from scratch or provide it with the necessary resources and code to run pipelines and deploy pipelines. Also, in addition to the working mode, Pangea allows the creation of a local environment for testing and benchmarking models.

The work [33] provides a detailed overview of the applicability of server technologies and tool platforms for implementing MLOps projects. It discusses the advantages that they can provide to the machine learning pipeline. The authors identified nine serverless platforms studied in MLOps works; among them, the AWS Lambda platform was the leader in the number of mentions. The authors pointed out the main problems of using serverless platforms in machine learning projects: the cost, ensuring a sufficient number of scalable resources, and the problem with output latency.

The paper [34] provides an overview of the fully customizable end-to-end SOLIS methodology. This methodology provides easy deployment and configuration of machine learning pipelines, allowing even non-specialists to participate in designing and delivering models to end users. The proposed architecture is scalable and allows using several tensor computing platforms, such as Tensorflow and Pytorch, to deploy neural network models. The stages of the SOLIS pipeline are fully customizable, so the project is not limited to any specific area and has a wide range of applications.

In [35], the CodeReef open framework is proposed for exchanging components necessary to provide cross-platform MLOps in non-virtualized, portable, configurable, and reusable open-source packages. The work aims to support the initiatives of MLOps communities on reproducibility, automate machine learning tests, and ensure the portability of MLOps models based on real cases.

c. Practical research

One of the interesting practice-oriented works is [36], which proposes a digital twin of a smart home integrated with MLOps and provides an adequate energy consumption prediction. To predict hourly electricity consumption, models from the XGBoost library are used, which are trained on consumption characteristics for the last 23, 24, and 25 hours. After training the models, the one with the most minor root-mean-square error for each combination of hyperparameters is selected. The digital twin integrated with machine learning models was successfully tested on a residential building equipped with IoT smart meters and smart sockets for 19 months, with measurements performed every second.

In [37], an MLOps pipeline is proposed to predict the market price of electricity using an artificial neural network. The pipeline includes a version control system, a machine learning system, testing services, model services, function storage, and deployment services. When building the pipeline, the authors used PyTorch as a machine learning library, Python as a scripting language, Jenkins as a continuous integration server, and Django as a web interface development environment. The proposed MLOps methodology was then generalized and successfully applied to two more cases.

In [38], the MLOps methodology is considered from the perspective of developing an interface that provides greater transparency, fairness, explainability, and continuous monitoring for machine learning models. The authors provide the Dales tool, a Python library that imple-

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

ments an interface independent of the machine learning model for interactive explainability and fairness. The interface is designed considering the functionality of various explicable machine learning tools; thus, it is aimed at unifying existing solutions.

Authors of [39] describe VMware's experience implementing a machine learning project to detect and diagnose performance problems in enterprise solutions deployed in a production environment. Firstly, a pipeline was created for continuous training and maintenance, which helps to update models in a few hours, which took the company about six months. Secondly, a monitoring module was implemented to visualize performance anomalies and form a user's idea of how machine learning models work. Thirdly, new models and performance diagnostics were automated to be delivered in a dynamically changing environment. Fourth, an experimental environment has been set up to test the behavior of the model, which allows us to assess how well the models respond to specific production scenarios.

The authors of [40] and [41] describe the practice of developing a medical software product, Oravizio, designed to assess the risks of joint replacement operations. This product has come a long way, from machine learning experiments to certification by all regulatory requirements for medical machine learning systems. Although Oravizio was developed and implemented before the emergence of the MLOps concept, its design goals align with the characteristics of MLOps, making it a good example of continuous software development in machine learning projects.

Finally, [42] and [43] are works whose subject matter is entirely relevant to extracting opinions from social media we are investigating. The authors [42] design a pipeline of DataOps/MLOps operations performed on data automatically extracted from Twitter to analyze user reviews of tourist services in Italy. The authors describe in detail the pipeline components responsible for data collection and preprocessing, thematic modeling, analysis of social networks, and classification of tweets but do not disclose the mechanisms of linking these components into a single pipeline. In this sense, [43] contains more technical details of implementing sentiment analysis based on MLOps. The paper describes the application of the Amazon Web Services ecosystem, starting with the annotation of training data using Ground Truth and ending with the deployment of the model using MLflow SageMaker.

Methodology

The architecture of the Astana Opinion Mining macro service is proposed and conceptualized as a cloud-based microservice structure encompassing four distinct yet interconnected pipelines. Each pipeline, represented as a series of microservices, serves a specific function within the overall system, as illustrated in Figure 2. The initial stage of our process is the data collection pipeline. While termed a 'pipeline,' this stage is more accurately a collection of loosely interconnected components, predominantly consisting of crawlers that operate both asynchronously and autonomously. However, its designation as a pipeline is forward-looking and will be described below. Following this, the Feature Engineering pipeline is dedicated to preprocessing data and preparing it for further analysis. Subsequently, the Experimentation pipeline is engaged in developing a machine-learning model tailored to the specific requirements of the task, which, in this instance, is sentiment analysis. The final stage, the Deployment and Maintenance pipeline, synthesizes the outputs from the preceding pipelines to deliver a comprehensive solution to the end user. This entire process is underpinned by a cloud-based microservice architecture, notable for its flexibility, ease of scalability, and the autonomy afforded to each microservice.

DOI: 10.37943/21CPQX5616

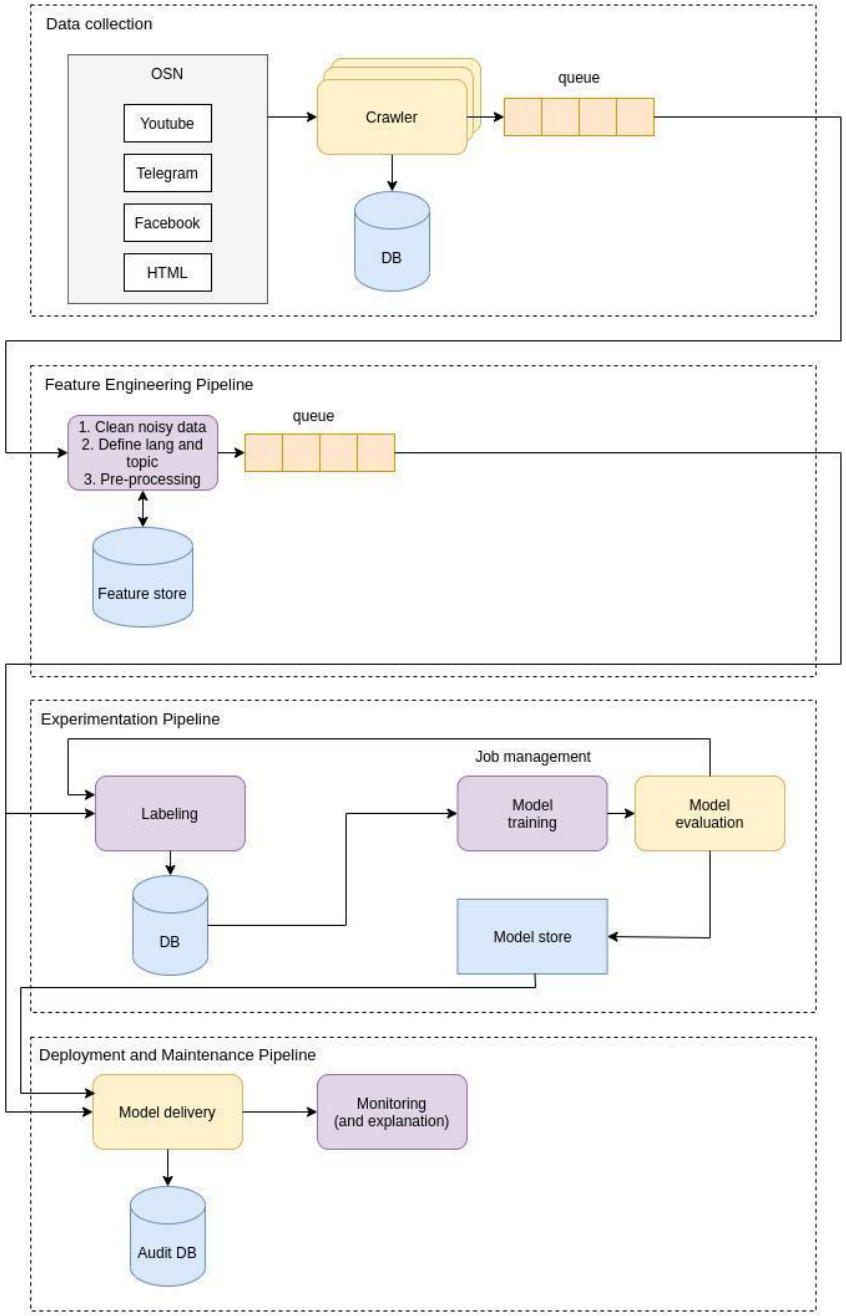


Figure 2. Proposed MLOps architecture.

Generally, the proposed architecture can be as following method presented step by step below:

Step 1: Data Collection Pipeline:

1.1. Collect text data from YouTube, Telegram, Facebook, and HTML sources using crawlers.

1.2. Store the collected data in a database.

1.3. Add the data to a processing queue.

Step 2: Feature Engineering Pipeline

2.1. Fetch data from Queue.

2.2. Clean the noisy data.

2.3. Define the language and topic of each message.

2.4. Preprocess the text.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

2.5. Store processed features in the Feature Store.
2.6. Push processed data back into the Queue for experimentation.
Step 3: Experimentation Pipeline
3.1. Fetch processed features from Queue.
3.2. Label data for supervised learning tasks.
3.3. Store labeled data into a Database (DB).
3.4. Train machine learning model using labeled data.
3.5. Evaluate model performance.
3.6. Store trained model in the Model Store.
3.7. Based on evaluation, optionally go back to Step 3.2. for improvements.
Step 4: Deployment and Maintenance Pipeline
4.1. Deliver trained model from the Model Store to production.
4.2. Store deployment audit logs in the Audit DB.
4.3. Monitor model performance and provide explanations for predictions.

Each of these pipelines presented step-by-step operates within the overarching cloud-based microservice architecture. This approach provides notable advantages, including enhanced flexibility, scalability, and independence for each microservice. Such a configuration allows for easy adaptation and growth in response to evolving data requirements and technological advancements.

a. Data Collection pipeline

Contrary to a traditional pipeline, this component is characterized by loosely coupled elements, primarily crawlers, functioning asynchronously and autonomously. This design choice emphasizes the system's modular nature, allowing for independent operation and flexibility in data acquisition. The Astana Opinion Mining service considers external resources of different structures and purposes, such as YouTube video hosting, the Telegram social network, and a website with residents' appeals: https://aitu.city. The primary data for analysis are text data generated by the user. On YouTube, this is the text of the comment; on Telegram, this is the text of the message; and on the website http://aitu.city, this is the text of the appeal. Although only user-generated text is currently being utilized, all available data from external sources is being collected for potential inclusion in the general data model. This data can later be transferred to the general data model. For example, YouTube likes correlate to emoji reactions on Telegram. Each microservice stores the collected information in its local database and sends it to the next module through its queue. Data transformation from all sources must occur in the subsequent module. This will allow crawlers to be developed independently of other modules, and changes in the next module will not entail chain changes in the data collection module.

The results of the pipeline are full-data-model and general-data-model. The full-data model includes all social platform-specific meta-data tailored to each platform's unique data structure and content type. For Telegram, the model encompasses a comprehensive set of fields such as id, message, date, message_id, telegram_chat_id, telegram_user_id, as well as metadata about message status like deleted, deletion_date, views, and threading information including message_date, reply_to_message_id, and message_thread_id. This allows a detailed analysis of the platform's user interactions and message dynamics.

Similarly, YouTube's model is designed to capture rich video and user interaction data, including fields like id, video_id, text, user_id, user_name, viewer_rating, like_count, and timing details captured in published_at and updated_at. Additionally, it tracks engagement metrics such as total_reply_count and parent_id, which are crucial for understanding viewer interactions with the content.

DOI: 10.37943/21CPQX5616

For the local website aitu.city, the model is structured to hold key information relevant to user-generated content, encompassing id, message, user_name, and temporal data like publish_date. It also includes views, votes, location, and status, offering a comprehensive view of user engagement and content relevancy within the website's context.

The general model, designed to aggregate data across these platforms, is more streamlined. It focuses on the essential fields of message, date, and osn_name (online social network name). This model facilitates a unified view of the data, enabling cross-platform analysis and insights while retaining the essence of user-generated content across different digital environments.

While the data collection pipeline currently functions primarily as a set of connectors for data collection from diverse sources, its designation as a pipeline is forward-looking. We anticipate extending and enriching this module with additional submodules that will not only collect data but also transform it into an enriched general-data-model.

b. Feature Engineering pipeline

This segment of the architecture is dedicated to the preprocessing of data. It involves meticulous refinement of raw data, ensuring that the input for the subsequent stages is optimized for accurate analysis. The pipeline's design facilitates efficient processing, which is essential for handling large volumes of data. The functional architecture of the module is centered around a singular base microservice responsible for executing four distinct tasks integral to the system's operation.

The first initial task encompasses receiving messages from the antecedent pipeline, followed by their storage in the database. It is designed to operate without any supplementary processing during the message reception phase. This strategy is pivotal for ensuring the uninterrupted and efficient reception of data, thereby minimizing the potential for data loss and avoiding accumulating unprocessed messages in the queue. The operational protocol entails retrieving 10 messages per session from the queue for processing.

The second task involves a crucial filtering process to eliminate extraneous elements such as noise and spam from the data stream. A streamlined data cleaning methodology is employed, wherein messages containing predefined keywords are selectively removed. This process is resource-efficient and was developed following an in-depth analysis of the data, leading to the identification of a set of keywords and phrases (e.g., «https,» «link,» «candid photos») commonly associated with irrelevant messages. The inclusion criteria for these keywords are structured to ensure their effective exclusion from subsequent processing stages. Statistical data analysis revealed that approximately 2% of daily messages fall into the "noise" and spam category. Furthermore, an evaluation of message content established that messages comprising fewer than five words typically lack substantive information. Consequently, such messages were excluded from further analysis, accounting for approximately 40% of daily message volume. This approach, structured and methodical, is integral to maintaining the relevance and quality of data within the system, ensuring that only pertinent information is forwarded for subsequent processing stages.

The module's third task encompasses the preprocessing of text and the extraction of textual characteristics. This task, along with the second task, operates on a minute-by-minute basis within an independent thread. In each iteration, a batch of up to 100 unprocessed messages is retrieved from the database for processing. At the outset of data preprocessing, a critical step involves determining the language of each message. For language classification, the «nikitast/lang-classifier-roberta» model is utilized [44]. The preprocessing phase includes two primary processes: text cleaning and lemmatization. The removal of punctuation marks and stop words is executed using resources from the NLTK library, tailored for Kazakh, Russian, and English languages, and further enhanced by inputs from our laboratory experts [45]. For

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

lemmatization, the SnowballStemmer tool within the NLTK library is employed for English text [46], whereas for Russian, the pymystem3 Python module is used [47]. After these preprocessing steps, the module engages in the extraction of underlying topics from the messages. This extraction aligns with the thematic areas relevant to the Smart City project, encompassing domains such as transport, urban infrastructure, healthcare, and education. The non-negative matrix factorization (NMF) method is employed for this purpose. A critical component of this process involves the creation of a thematic dictionary for each Smart City topic. This dictionary, comprising keywords and concepts in Kazakh, Russian, and English, serves as the basis for isolating pertinent data. For instance, in the context of transport and urban infrastructure, a comprehensive dictionary is compiled, encompassing a wide array of terms pertinent to this domain: «buses», «trolleybuses», «trams», «metro», «trains», «minibuses», «taxi», «bicycles for rent», «car rental», «public transport», «municipal transport», « regular bus», «long-distance bus», «vehicle», «freight transport», «passenger transport», «transport network», «transport hub», «railway station», «bus station», «airport», «transport system», «infrastructure», «route», «schedule», «stop», «platform», «parking», «passenger», «driver», «conductor», «ticket», «travel ticket», «paid parking», «fare», «map», «free travel», «parking», «crossroads», «traffic flow», «traffic jams», «traffic», «speed limit», «pedestrian crossing», « traffic light», «pass», «road marking», «ring road», «overpass», «underpass». The application of the NMF (Non-negative Matrix Factorization) technique facilitates the extraction of transport-related themes from the text data, which has been preliminarily filtered using the specially compiled dictionary. Table 1 presents a synopsis of five distinct topics identified using the NMF method. The designation of each topic is derived from the analysis of the most contributive terms and keywords, as determined by their TF-IDF (Term Frequency-Inverse Document Frequency) weights.

In the realm of topic modeling, the first identified topic centers around the challenges associated with public transportation. The second topic delves into issues related to bus stop locations for fixed-route vehicles, emphasizing the public's need for sheltered waiting areas, particularly in inclement weather conditions. The third topic addresses the broader road infrastructure issues, highlighting scenarios where traffic flow is hindered or completely halted due to road closures. The fourth topic examines the quality of service, pricing, and convenience factors of online taxi services, including prominent providers like Yandex and Uber. The fifth topic explores the management and upkeep of parking facilities within the urban infrastructure. This systematic approach to topic identification extends across all analyzed messages. Table 2 illustrates a segment of the dataset that has been labeled following these identified themes.

Table 1. Topics identified by NMF

Keywords	Topic
'bus', 'route', 'minute', 'wait', 'why', 'drive', 'walk', 'ride', 'new', 'scheme', 'picture', 'change', 'interval', 'reduce', 'motion', 'application'	Problems of public transport
'stop', 'warm', 'supply', 'ride', 'stand', 'warm', 'whole', 'freeze', 'station', 'near', 'city', 'do', 'reach', 'regular', 'poke', 'clean'	The need for warm bus stops
'traffic', 'road', 'day', 'turan', 'why', 'stand', 'overlap', 'know', 'little', 'street', 'ride', 'big', 'concert', 'appear', 'ask', 'airport'	Traffic jams and other problems on the road
'taxi', 'yandex', 'work', 'order', 'drive', 'uber', 'airport', 'price', 'expensive', 'driver', 'expensive', 'touch', 'fare', 'show', 'taxi driver', 'cheap'	Taxi prices
'parking', 'paid', 'place', 'driver', 'courtyard', 'whole', 'pay', 'city', 'clean up', 'child', 'in general', 'akimat', 'crossing', 'car', 'find', 'free'	Parking lots

DOI: 10.37943/21CPQX5616

The final step in the processing sequence involves systematically recording several key elements within the microservice database. This includes the original message, its processed form, the identified language of the message, and the ascertained topic based on the sample analysis. Additionally, the database records the date on which each message underwent processing.

In the concluding task of this stage, the model, which encapsulates the original message, its preprocessed version, the determined language, and the identified topic, is dispatched to two distinct queues. These queues serve as conduits, channeling the processed data to the subsequent stages in the pipeline. This methodical progression ensures a seamless flow of information through the different phases of the system.

Table 2. Examples of messages and their identified topics.

No	Comments	Topic
1	“How cool I don't need to use a taxi, I use bike sharing, but it's economical”	Taxi prices
2	“The business center was chosen very unsuccessfully, there are practically no parking lots”	Parking lots
3	“If there are no regular stops, then what can we say about warm stops. Again, at minus 30 to stand in the cold”	The need for comfortable bus stops
4	“Taxi drivers already pay taxes to the platform that gave them the opportunity to work. And what are they asking to pay for? They pay tax for cars, they pay for roads, they pay fines, they pay for gasoline, they pay for the Internet on the phone. Make some of this free or at least cheaper”	Problems of public transport
5	“Guys, which air-conditioned buses are going towards the beginning of Seifullin street.”	The need for comfortable bus stops
6	“Summer traffic jams: roads do. Traffic jams in winter: snow is being cleaned. Spring traffic jams: everything melts. Traffic jams in the fall: school starts)) It's just me, thoughts out loud))”	Traffic jams and other problems on the road

c. Experimental pipeline

This pipeline is central to the development of the machine learning model and the crux of our sentiment analysis task. It encompasses the rigorous process of model creation, from initial experimentation to fine-tuning, ensuring that the model is tailored to the specific requirements of sentiment analysis within the smart city context.

The experimental pipeline comprises three primary modules: data labeling, model training and evaluation, and model storage.

The data labeling module is at the forefront of the pipeline, equipped with automated capabilities to ensure a consistent influx of data from established sources. This module is designed with an interactive interface, enabling users to label incoming data selectively. The versatility of the module’s interface allows for easy transition among various annotation tasks. In sentiment analysis, for instance, each message is labeled None, Positive, Negative, or Neutral. For tasks like message classification, the interface offers predefined categories like «Relevant,» «Noise,» and «Spam» for user selection. This dynamic and user-friendly interface of the data labeling module significantly enhances the efficiency of the annotation process, which is vital for the quality of data used in subsequent model training and evaluation stages.

The current approach involves manual data labeling, which is resource-intensive but provides high-quality labeled data. To optimize this, semi-supervised learning approaches could be applied. These methods use a small amount of labeled data and a larger amount of unlabe-

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

led data. This can significantly reduce the human effort needed for labeling without substantially compromising the quality of the training data.

The subsequent element of this pipeline is the model training module, which operates by receiving Python code files through a REST API to construct a machine learning model. This file is uploaded by a machine learning engineer and must adhere to specific criteria for the code to be viable:

1. Training data must be sourced from the database containing labeled data.
2. The model in training should be configured to process textual input.
3. Post-training, the model needs to be stored as a binary file.
4. In the context of sentiment analysis, the model should categorize data into three classes: positive, negative, and neutral.

The REST API integrates several critical parameters:

- *Code Expiry:* This parameter defines the duration for which the code remains executable.
- *Execution Frequency:* Considering that data labeling is not instantaneous, initial model training might not yield optimal results. Therefore, establishing a regular schedule for code execution, such as weekly activation every Saturday morning for a month, is recommended.
- *Accuracy Threshold*: This is set based on the desired quality of the model. Only models surpassing this accuracy benchmark proceed to the next stage.

Before the model is archived, it undergoes an evaluation phase. Models that demonstrate accuracy above the predefined threshold are then stored in the Model Store component. The resultant model is saved as a binary file in the Model Store, which is structured as a git repository to facilitate model versioning. This setup allows subsequent pipelines to efficiently retrieve the model directly from the repository.

d. Deployment and Maintenance pipeline

The Deployment and Maintenance pipeline is the culmination of the process, where the data refined by previous pipelines and the sentiment analysis model are put into action. This pipeline primarily handles applying the model to a continuous stream of processed messages from various social networks.

Central to this pipeline is the model delivery microservice. Its key function is to retrieve the most up-to-date model from the model store and implement it on incoming data from the Feature Extraction pipeline. The outcomes, specifically the sentiment classifications of the messages, are then systematically recorded in the database. This record includes the sentiment results and annotations regarding which model generated these results.

The Apache Superset platform [48] is employed for monitoring and oversight purposes. Hosted on our servers, Superset is integrated with the local databases of all the microservices, offering a comprehensive and adaptable monitoring solution. The dashboard of the Superset platform is customized to display pertinent information, providing a clear and detailed overview of the system's performance and outputs.

Results and Discussion

This section presents the results of the implemented architecture. While architecture was successfully implemented the proposed, it is worth noting that model training requires significant resources. As a result, a Python 3 server accelerator was employed based on Google Compute Engine (GPU). In future work, we plan to connect our servers equipped with graphics processors.

DOI: 10.37943/21CPQX5616

a. Architecture Implementation

All microservices projects are under the git version control tool with common settings and a common CI/CD. Continuous Integration (CI) helps build and test the code before its direct deployment to production. Continuous deployment (CD) with microservice architecture allows quick software solutions delivery to the customer.

Our end-user application was built using the IDEAL paradigm, which fits the data collection and processing system. IDEAL means that the application should be [Isolated state] isolated, [Distributed] have a distributed architecture, be [Elastic] flexible in the sense of horizontal scaling, be controlled by [Automated] automated systems, and its components should be [Loose coupling] loosely coupled [49]. In our architecture, we used the following technologies:

- Docker Swarm allows to create a fault-tolerant, scalable, easily and quickly recoverable system. Automating the recovery of containers after a failure, due to health-checking, provides fault tolerance. Due to the function of container replication, horizontal scaling of applications is achieved in accordance with the load. All microservices are managed by Docker Swarm and can be seen via web-interface runned by Portainer framework [50] (see Figure 3).
- RabbitMQ is used to create queues between microservices [51]. Queues give us the ability to process huge amounts of data asynchronously, consistently, and fault-tolerantly (See Figure 4).
- Traefik gives the ability to manage http traffic within the docker swarm ingress network.
- OpenStack manages resources and virtual machines, which greatly simplifies the work with server hardware.

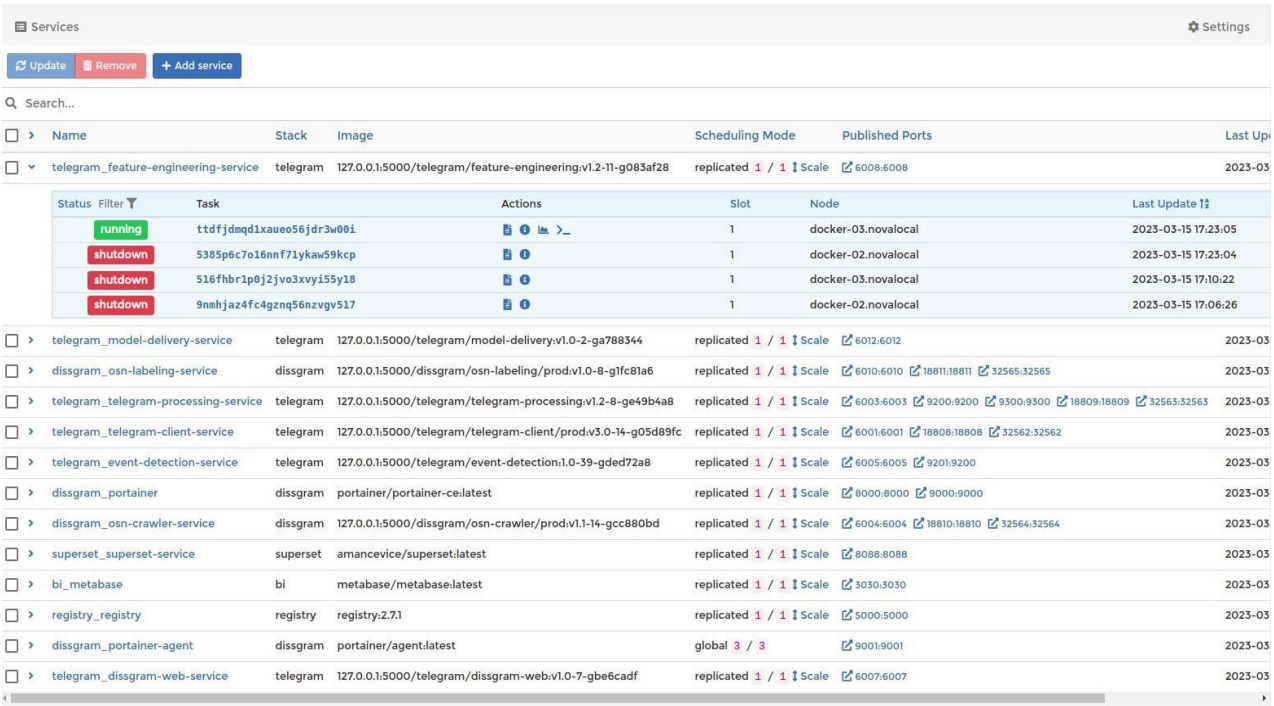
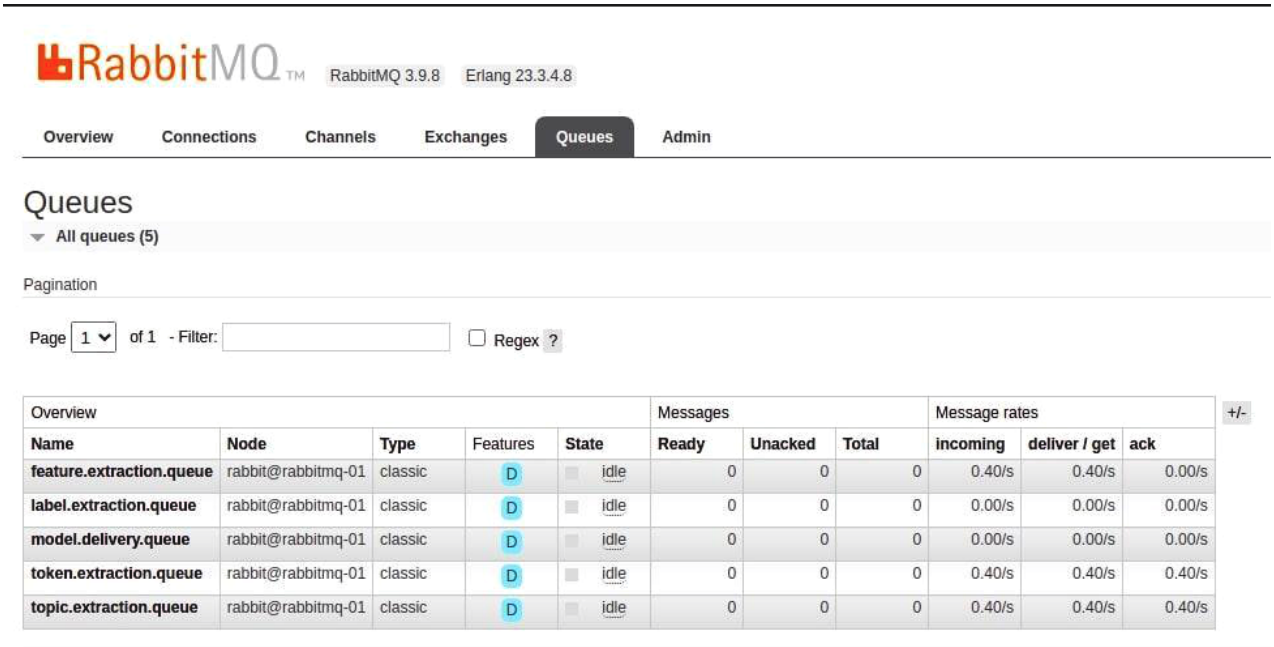


Figure 3. Screenshot from Portainer web user interface for Docker Swarm.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025



Overview					Messages			Message rates			+/-
Name	Node	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
feature.extraction.queue	rabbit@rabbitmq-01	classic	D	idle	0	0	0	0.40/s	0.40/s	0.00/s	
label.extraction.queue	rabbit@rabbitmq-01	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
model.delivery.queue	rabbit@rabbitmq-01	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
token.extraction.queue	rabbit@rabbitmq-01	classic	D	idle	0	0	0	0.40/s	0.40/s	0.40/s	
topic.extraction.queue	rabbit@rabbitmq-01	classic	D	idle	0	0	0	0.40/s	0.40/s	0.40/s	

Figure 4. Screenshot of RabbitMQ admin web user interface.

The technical side of our microservices:

- Data collection modules are written in Java and use public social networking APIs. For example, Telegram provides access to its TDLib library, with which you can create a bot-user and include it in public Telegram channels [52]. Youtube provides an API with Google Key authorization for parsing videos and comments. Not all information about comments is available. For example, only the content owner could see the number of dislikes on the video. The website crawling does not require additional APIs, it is enough to download pages and define the parsing algorithm for a particular website.
- The Data Labeling module is written in Java. Since this microservice works only with the incoming stream of data it needs a simple designed UI for data labeling specialists. For the native UI, the Vaadin open-source web application development platform is used [53].
- Modules in the Feature Engineering pipeline and modules for model training, evaluation and delivery are written in Python, because it is more lightweight and has convenient libraries for data processing and machine learning.
- PostgreSQL was used as a database in each pipeline.
- For the current stage of our work the capabilities of the Apache Superset tool are sufficient to represent the data in all pipelines of the architecture.

The monitoring board consists of data statistics and actual data example parts. The data statistics part presents the number of messages in pipelines and number of messages according to social networks (see Figure 5). The amount of messages according to the criteria for sustainable development of the city (see Figure 6) and according to the sentiment (see Figure 7) are presented in the diagrams.

The source code for the crawler designed for Telegram can be found within the GitHub project located at https://github.com/orgs/knowledge-extraction-system/repositories. This project is expected to expand over time to include the code for additional microservices.

DOI: 10.37943/21CPQX5616

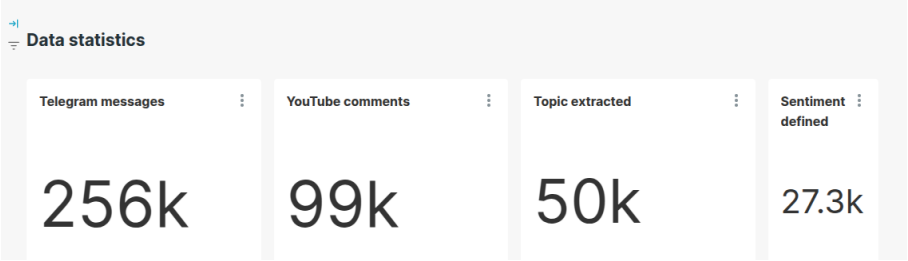


Figure 5. Apache Superset screenshot of the monitoring dashboard. Data statistics.

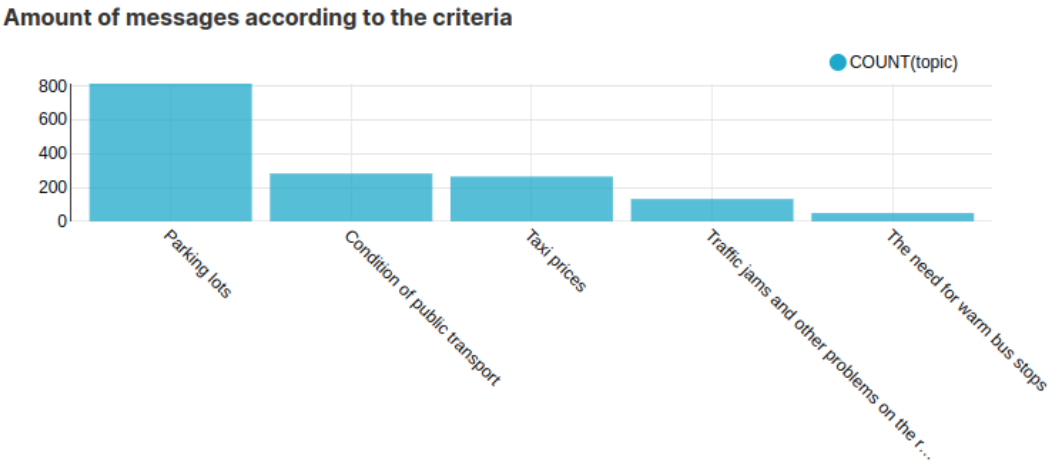


Figure 6. Apache Superset screenshot of the number of messages according to the criteria.

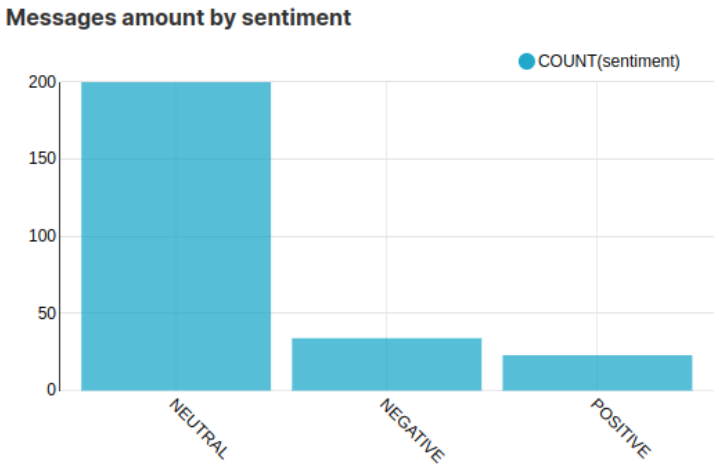


Figure 7. Apache Superset screenshot of the messages amount by sentiment.

b. Sentiment analysis

The sentiment analysis task is not new and already has solutions for different languages. However, as the Kazakh language is a low-resource language [54], large language models (LLMs) must be improved. Our approach, focusing on task-specific models, demands significantly fewer training resources than LLMs' extensive requirements. Unlike LLMs, which are trained on massive datasets demanding high computational power and storage, our models utilize a smaller, more specialized dataset, reducing computational load and energy consumption. This aspect is particularly beneficial in the framework of MLOps, where managing resources efficiently is crucial. Moreover, integrating these lighter, specialized models into a fully automated MLOps pipeline enhances manageability and cost-effectiveness. It enables quicker training and deployment cycles, which is essential for the dynamic nature of sentiment anal-

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

ysis tasks. In contrast, the integration of LLMs into such pipelines, although powerful in their capabilities, can pose challenges due to their size and complexity, especially in scenarios requiring continuous training and deployment. This presents a trade-off between the broad capabilities of LLMs and the practicality and resource efficiency of specialized models, underscoring the importance of aligning model choice with the specific requirements and resource constraints of the sentiment analysis task in smart city applications.

The implemented solution was divided into two branches: the Kazakh and Russian languages. In this study, we will delve into the reviews and opinions citizens express on transportation. However, our approach extends beyond a narrow scope and encompasses a broader perspective. As mentioned, we employ the SULPITER methodology to gather and analyze reviews and opinions about smart city indicators. The proposed pipeline allows us to collect data on various aspects of smart city indicators, including but not limited to transportation. This flexibility is further enhanced by the Model Delivery module, which enables us to fine-tune our model to incorporate and analyze other relevant indicators.

For the Kazakh language, the pre-trained multilingual XLM Roberta model has been fine-tuned [55]. The model was fine-tuned on labeled data in the Kazakh language obtained from the Data Labeling database. After the first training, eight epochs were passed on data in the amount of 1091 messages, which consisted of 339 neutral, 328 positive, and 424 negative samples. The dataset was divided into train and test subsets that contain 873 and 218 records, respectively. After fine-tuning, the training accuracy was equal to 79.2%, and accuracy on the test data showed 68.97%. It is not enough for deploying in practical applications. Data augmentation was performed for that point, and the updated dataset comprised 6999 messages. The labeled dataset consists of 2170 neutral, 2450 positive, and 2379 negative samples. The accuracy of the newly trained model on test data is 83.71%. Training accuracy statistics are presented in Figure 8. Here, we see that the model overtrained after the fourth epoch and obtained 96.73% accuracy after eight epochs of fine-tuning.

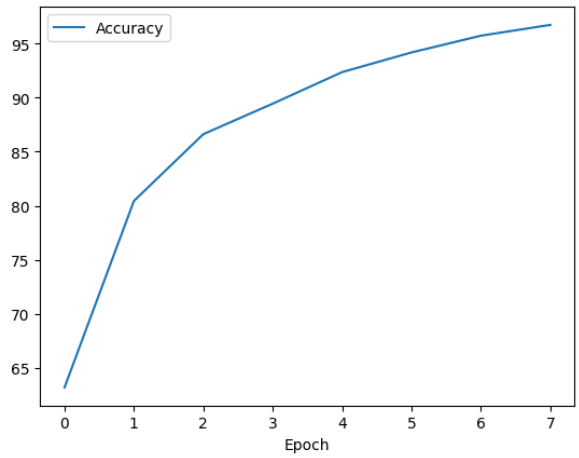


Figure 8. Fine-tuning of XLM RoBERTa model on the sentiment dataset.

During the experiment, four models of sentiment analysis were tested in Russian: «Blanchefort», «Sismetanin», «MonoHime» and «Dostoevsky». The goal was to evaluate the effectiveness of these models and choose the most suitable one for sentiment analysis. It is important to note that each model uses its own classification system, which could lead to a loss of accuracy when comparing the results. For example, the Dostoevsky model has five classification classes, while other models operate with three classes. To ensure comparability of the results and eliminate the problem of different classification systems, a module was developed that allows to translate gradations and classify texts into three classes: negative (2), positive (1)

DOI: 10.37943/21CPQX5616

and neutral (0). A comparative analysis revealed that the Blanchefort model showed the best accuracy of sentiment analysis (see Table 3). It achieved an accuracy of 71%, which confirms the expected range of results of baseline models of deep transfer learning for sentiment analysis in Russian, ranging from 70% to 76% [56].

Table 3. Comparison of sentiment analysis scores for four models.

F1 score	Accuracy	Model
0.72	71.43%	Blanchefort
0.51	54.55%	Sismetanin
0.59	59.74%	MonoHime
0.46	46.75%	Dostoevsky

Thus, the results obtained confirm that the «Blanchefort» model is the most effective for this study. There is still room for further investigations to explore and improve the quality and precision of the language models. Using recently available datasets (e.g. the largest Russian-language dataset of reviews [57]) or collecting our own we can elaborate on improving the performance of sentiment analysis in smart city applications which is crucial for gaining the trust of public authorities and stakeholders.

To calculate the Sentiment-based Perception Score (P), we follow a formula for calculating the Perception Score:

$$P = \frac{PositiveSentimentCount - NegativeSentimentCount}{TotalSentimentCount} \quad (1)$$

where Positive and Negative Sentiment Counts are the numbers of instances where sentiment analysis determines a positive or negative sentiment respectively in the citizen feedback or data; Total Sentiment Count is the total number of instances where sentiment analysis was performed on the citizen feedback or data. The result will be a score ranging from -1 to 1, where a positive score indicates a predominantly positive perception, a negative score indicates a predominantly negative perception, and a score close to zero suggests a neutral or mixed perception. Then, the Sentiment-based Perception Score is included as a component within a comprehensive formula for evaluating the sustainable development of a smart city. This formula incorporates empirically determined weights assigned to each indicator.

Conclusion

Using the MLOps methodology, we successfully implemented a cloud-based microservices architecture for the Astana Opinion Mining macro-service. The architecture was customized to support various tasks and infrastructure facilities based on a taxonomy of criteria for sustainable development in the urban environment. The crawled data from external resources undergo processing, validation, and classification based on the taxonomy criteria via the NLP pipeline. The resulting aspect-oriented polarities are extracted using a well-trained Opinion Mining (Sentiment Analysis) model. We have recently incorporated Topic Modeling as a simple yet effective approach to extracting relevant aspects. In our future work, we plan to enhance our capabilities by developing models for aspect-based sentiment analysis (ABSA) using a unique ABSA dataset specifically curated for city problems, enabling a more fine-grained analysis of citizen sentiments and opinions. This will further enrich our understanding of urban challenges and support targeted interventions for sustainable urban development.

Thus, the resulting solution has practical applications, allowing for real-time analysis of the city's situation. The architecture is easily scalable, fault-tolerant, and automated. With its

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

ability to process, validate, and classify data based on a taxonomy of sustainable development criteria and extract aspect-oriented polarities through advanced sentiment analysis, the solution offers real-time insights and analysis of the urban environment.

Acknowledgement

This research has been funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No.BR24992852 "Intelligent models and methods of Smart City digital ecosystem for sustainable development and the citizens' quality of life improvement")

References

[1] Borg, M. (2022). Agility in Software 2.0 – Notebook Interfaces and MLOps with Buttresses and Rebars. In *Lecture notes in business information processing* (pp. 3–16). https://doi.org/10.1007/978-3-030-94238-0_1

[2] Bernardi, L., Mavridis, T., & Estevez, P. (2019). 150 Successful Machine Learning Models: 6 lessons learned at booking.com. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1743–1751. https://doi.org/10.1145/3292500.3330744

[3] Bosch, J., Olsson, H. H., & Crnkovic, I. (2021). Engineering ai systems: A research agenda. Artificial intelligence paradigms for smart cyber-physical systems, 1-19. https://doi.org/10.4018/978-1-7998-5101-1.ch001

[4] Testi, M., Ballabio, M., Frontoni, E., Iannello, G., Moccia, S., Soda, P., & Vessio, G. (2022). MLOps: A Taxonomy and a Methodology. IEEE Access, 10, 63606–63618. https://doi.org/10.1109/access.2022.3181730

[5] Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying DevOps Practices of Continuous Automation for Machine Learning. Information, 11(7), 363. https://doi.org/10.3390/info11070363

[6] Rubini, L., & Della Lucia, L. (2018). Governance and the stakeholders' engagement in city logistics: the SULPiTER methodology and the Bologna application. Transportation Research Procedia, 30, 255–264. https://doi.org/10.1016/j.trpro.2018.09.028

[7] Witanto, J.N., Lim, H., & Atiquzzaman, M. (2018). Smart government framework with geo-crowdsourcing and social media analysis. Future Generation Computer Systems, 89, 1–9. https://doi.org/10.1016/j.future.2018.06.019

[8] Lin, Y., & Geertman, S. (2019). Can social media play a role in urban planning? A literature review. Computational Urban Planning and Management for Smart Cities 16, 69-84.

[9] Steils, N., Hanine, S., Rochdane, H., & Hamdani, S. (2021). Urban crowdsourcing: Stakeholder selection and dynamic knowledge flows in high and low complexity projects. Industrial Marketing Management, 94, 164–173. https://doi.org/10.1016/j.indmarman.2021.02.011

[10] Alizadeh, T. (2018, May). Crowdsourced smart cities versus corporate smart cities. In IOP conference series: Earth and environmental science (Vol. 158, No. 1, p. 012046). IOP Publishing.

[11] Ilieva, R.T., & McPhearson, T. (2018). Social media data for urban sustainability. Nature Sustainability, 1(10), 553–565. https://doi.org/10.1038/s41893-018-0153-6

[12] Schrammeijer, E. A., Van Zanten, B. T., & Verburg, P. H. (2021). Whose park? Crowdsourcing citizen's urban green space preferences to inform needs-based management decisions. Sustainable Cities and Society, 74, 103249. https://doi.org/10.1016/j.scs.2021.103249

[13] Ghermandi, A., & Sinclair, M. (2019). Passive crowdsourcing of social media in environmental research: A systematic map. Global Environmental Change, 55, 36–47. https://doi.org/10.1016/j.gloenvcha.2019.02.003

[14] Mcardle, G., & Kitchin, R. (2016). Improving the Veracity of Open and Real-Time Urban Data. Built Environment, 42(3), 457–473. https://doi.org/10.2148/benv.42.3.457

[15] Palladini, A. (2022). Streamline machine learning projects to production using cutting-edge MLOps best practices on AWS (Doctoral dissertation, Politecnico di Torino).

[16] Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. IEEE Access, 11, 31866–31879. https://doi.org/10.1109/access.2023.3262138
[17] Van Den Heuvel, W., & Tamburri, D.A. (2020). Model-Driven ML-Ops for Intelligent Enterprise Applications: Vision, Approaches and Challenges. In Lecture notes in business information processing (pp. 169–181). https://doi.org/10.1007/978-3-030-52306-0_11
[18] Renggli, C., Rimanic, L., Gurel, N. M., Karlas, B., Wu, W., & Zhang, C. (2021). A Data Quality-Driven View of MLOps. IEEE Data(Base) Engineering Bulletin, 44(1), 11–23. https://www.microsoft.com/en-us/research/publication/a-data-quality-driven-view-of-mlops/
[19] Karimi, M. R., Gürel, N. M., Karlaš, B., Rausch, J., Zhang, C., & Krause, A. (2021, March). Online active model selection for pre-trained classifiers. In International Conference on Artificial Intelligence and Statistics (pp. 307-315). PMLR.
[20] Renggli, C., Karlaš, B., Ding, B., Liu, F., Schawinski, K., Wu, W., & Zhang, C. (2019). Continuous integration of machine learning models with ease. ml/ci: Towards a rigorous yet practical treatment. Proceedings of Machine Learning and Systems, 1, 322-333.
[21] Renggli, C., Rimanic, L., Kolar, L., Hollenstein, N., Wu, W., & Zhang, C. (2020). On automatic feasibility study for machine learning application development with ease. ml/snoopy. arXiv preprint arXiv:2010.08410.
[22] Moreschini, S., Lomio, F., Hastbacka, D., & Taibi, D. (2022). MLOps for evolvable AI intensive software systems. 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 1293–1294. https://doi.org/10.1109/saner53432.2022.00155
[23] Mucha, T.M., Ma, S., & Abhari, K. (2022, August). Beyond MLOps: The Lifecycle of Machine Learning-based Solutions. In AMCIS.
[24] Matsui, B.M., & Goya, D.H. (2022, May). MLOps: A Guide to its Adoption in the Context of Responsible AI. In Proceedings of the 1st Workshop on Software Engineering for Responsible AI (pp. 45-49).
[25] Zhao, Y. (2021). Machine learning in production: A literature.
[26] Ruf, P., Madan, M., Reich, C., & Ould-Abdeslam, D. (2021). Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools. Applied Sciences, 11(19), 8861. https://doi.org/10.3390/app11198861
[27] Hewage, N., & Meedeniya, D. (2022). Machine learning operations: A survey on MLOps tool support. arXiv preprint arXiv:2202.10169.
[28] Recupito, G., Pecorelli, F., Catolino, G., Moreschini, S., Di Nucci, D., Palomba, F., & Tamburri, D. A. (2022). A Multivocal Literature Review of MLOps Tools and Features. In 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 84–91. https://doi.org/10.1109/seaa56994.2022.00021
[29] Zhao, Y., Belloum, A. S., & Zhao, Z. (2022). Mlops scaling machine learning lifecycle in an industrial setting. International Journal of Industrial and Manufacturing Engineering, 16(5), 138-148.
[30] Raj, E., Buffoni, D., Westerlund, M., & Ahola, K. (2021). Edge MLOps: An Automation Framework for AIoT Applications. In 2021 IEEE International Conference on Cloud Engineering (IC2E), 191–200. https://doi.org/10.1109/ic2e52221.2021.00034
[31] Antonini, M., Pincheira, M., Vecchio, M., & Antonelli, F. (2022). Tiny-MLOps: a framework for orchestrating ML applications at the far edge of IoT systems. In 2022 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), 1–8. https://doi.org/10.1109/eais51927.2022.9787703
[32] Miñón, R., Diaz-De-Arcaya, J., Torre-Bastida, A. I., & Hartlieb, P. (2022). Pangea: An MLOps Tool for Automatically Generating Infrastructure and Deploying Analytic Pipelines in Edge, Fog and Cloud Layers. Sensors, 22(12), 4425. https://doi.org/10.3390/s22124425
[33] Barrak, A., Petrillo, F., & Jaafar, F. (2022). Serverless on Machine Learning: A Systematic Mapping Study. IEEE Access, 10, 99337–99352. https://doi.org/10.1109/access.2022.3206366
[34] Ciobanu, R., Purdila, A., Piciu, L., & Damian, A. (2021). SOLIS--The MLOps journey from data acquisition to actionable insights. arXiv preprint arXiv:2112.11925.
[35] Garg, S., Pundir, P., Rathee, G., Gupta, P., Garg, S., & Ahlawat, S. (2021). On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps. In 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 25–28. https://doi.org/10.1109/aike52691.2021.00010

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 21, MARCH 2025

[36] Fujii, T.Y., Hayashi, V.T., Arakaki, R., Ruggiero, W.V., Bulla, R., Hayashi, F.H., & Khalil, K.A. (2021). A Digital Twin Architecture Model Applied with MLOps Techniques to Improve Short-Term Energy Consumption Prediction. Machines, 10(1), 23. https://doi.org/10.3390/machines10010023
[37] Subramanya, R., Sierla, S., & Vyatkin, V. (2022). From DevOps to MLOps: Overview and Application to Electricity Market Forecasting. Applied Sciences, 12(19), 9851. https://doi.org/10.3390/app12199851
[38] Baniecki, H., Kretowicz, W., PiÄ, P., & WiĹ, J. (2021). Dalex: responsible machine learning with interactive explainability and fairness in python. Journal of Machine Learning Research, 22(214), 1-7.
[39] Banerjee, A., Chen, C., Hung, C., Huang, X., Wang, Y., & Chevesaran, R. (2020). Challenges and Experiences with MLOps for Performance Diagnostics in Hybrid-Cloud Enterprise Software Deployments. In 2020 USENIX Conference on Operational Machine Learning (OpML 20). https://www.usenix.org/system/files/opml20-paper-banerjee.pdf
[40] Granlund, T., Stirbu, V., & Mikkonen, T. (2021). Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product. SN Computer Science, 2(5). https://doi.org/10.1007/s42979-021-00726-1
[41] Granlund, T., Kopponen, A., Stirbu, V., Myllyaho, L., & Mikkonen, T. (2021, May 1). MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases. https://oraviz.io/, 82–88. https://doi.org/10.1109/wain52551.2021.00019
[42] Stirparo, D., Penna, B., Kazemi, M., & Shashaj, A. (2022). Mining tourism experience on Twitter: a case study. arXiv preprint arXiv:2207.00816.
[43] Olsen, R., Ahmed, N., & Alekseev, I. (2022, April 4). Build an MLOps sentiment analysis pipeline using Amazon SageMaker Ground Truth and Databricks MLflow. https://aws.amazon.com. Retrieved September 2, 2023, from https://aws.amazon.com/ru/blogs/machine-learning/build-an-mlops-sentiment-analysis-pipeline-using-amazon-sagemaker-ground-truth-and-databricks-mlflow/
[44] Stepanov, N. (2023, January 7). Nikitast/lang-classifier-roberta. https://huggingface.co. Retrieved March 10, 2023, from https://huggingface.co/nikitast/lang-classifier-roberta#roberta-for-single-language-classification
[45] NLTK. (2023, January 2). https://www.nltk.org/. Retrieved February 15, 2023, from https://www.nltk.org/
[46] Documentation nltk.stem.snowball module. (2023, January 2). https://www.nltk.org. Retrieved February 15, 2023, from https://www.nltk.org/api/nltk.stem.snowball.html#module-nltk.stem.snowball
[47] Sukhonin, D., & Panchenko, A. (2018, July 4). PYMYSTEM3. https://pypi.org. Retrieved March 10, 2023, from https://pypi.org/project/pymystem3/
[48] The Apache Software Foundation. (n.d.). Apache Superset. https://superset.apache.org/. Retrieved March 10, 2023, from https://superset.apache.org/\
[49] Fehling, C., Leymann, F., Retter, R., Schupeck, W., & Arbitter, P. (2014). Cloud Computing Patterns. https://doi.org/10.1007/978-3-7091-1568-8
[50] Portainer.io (n.d.). Portainer. https://www.portainer.io/. Retrieved February 20, 2020, from https://www.portainer.io/
[51] Vmware (n.d.). RabbitMQ. https://www.rabbitmq.com/. Retrieved February 20, 2020, from https://www.rabbitmq.com/
[52] Telegram. (n.d.). Telegram database library. https://core.telegram.org/tdlib. Retrieved February 20, 2020, from https://core.telegram.org/tdlib
[53] Vaadin. (2022, June 28). https://vaadin.com/. Retrieved April 10, 2023, from https://vaadin.com/
[54] Haisa, G., & Altenbek, G. (2022). Multi-Task Learning Model for Kazakh Query Understanding. Sensors, 22(24), 9810. https://doi.org/10.3390/s22249810
[55] Singh, P., De Clercq, O., & Lefever, E. (2023). Distilling Monolingual Models from Large Multilingual Transformers. Electronics, 12(4), 1022. https://doi.org/10.3390/electronics12041022
[56] sismetanin/xlm_roberta_base-ru-sentiment-rureviews Hugging Face. (n.d.). https://huggingface.co/sismetanin/xlm_roberta_base-ru-sentiment-rureviews
[57] Yandex. (2023). Geo Reviews Dataset. https://github.com/yandex/. Retrieved October 14, 2023, from https://github.com/yandex/geo-reviews-dataset-2023