Ukrainian Big Data: The Problem Of Databases Localization

Victoria Vysotska[1], Ihor Shubin[2], Maksym Mezentsev[3], Karen Kobernyk[4] and Grygoryy Chetverikov[5]

[1] *Kharkiv National University of Radioelectronics, Nauky ave. 14, Kharkiv, 61166, Ukraine*
[2] *Lviv Polytechnic National University, Stepan Bandera Street, 12, Lviv, 79013, Ukraine*

Abstract
This paper explores the challenges surrounding the localization of Ukrainian databases within the realm of big data. It delves into the complexities of relative database localization. This research aims to recover the main problems of databases and big data localization of any kind so it could be used in potential solutions that people will chose suitable for them and to lay the groundwork for addressing the issue of big data localization in Ukrainian databases.

Keywords
Information retrieval, Machine translation, Databases, Database queries, Database localization.

1. Introduction

The proliferation of big data technologies has revolutionized the data management and analytics landscape, bringing both opportunities and challenges to a variety of industries. In the Ukrainian context, database localization has become a key challenge in maximizing the potential of big data analytics. As the volume and complexity of data grows rapidly, the need for effective localization strategies tailored to Ukrainian databases is becoming increasingly important.

In modern information systems, database localization refers to the process of adapting data storage and retrieval mechanisms to a specific geographic or linguistic context. Ukrainian databases, in particular, face hurdles unique to localization due to linguistic complexities, cultural nuances, and regulatory frameworks. This complexity requires a nuanced approach to database management[1] to ensure optimal performance and relevance in the Ukrainian context.

The significance of addressing the issue of relative database localization in Ukraine goes beyond mere technical considerations. It supports data integrity and accessibility, which are essential for informed decision-making, innovation, and social progress. By overcoming the challenges associated with database localization, Ukrainian stakeholders can extract valuable insights, promote economic growth, and facilitate technological progress in a variety of domains.

This paper aims to delve into the complex issues surrounding the localization of Ukrainian databases in the framework of Big Data. Through comprehensive analysis and research, we seek to shed light on the nature of the localization challenge and make it as a ground for the future solutions[2]. While recognizing the complexity of the task at hand, we aim to pave the way for future advances in database localization methodologies and practices tailored to the Ukrainian context.

COLINS-2024: 8th International Conference on Computational Linguistics and Intelligent Systems, April 12–13, 2024, Lviv, Ukraine

victoria.a.vysotska@lpnu.ua (V. A. Vysotska); igor.shubin@nure.ua (I. U. Shubin); maksym.mezentsev@nure.ua (M. A. Mezentsev); karen.kobernyk@nure.ua (K.S. Kobernyk); grigorij.chetverykov@nure.ua (G. G. Chetverykov)
0000-0001-6417-3689 (V. A. Vysotska); 0000-0002-1073-023X (I. U. Shubin); 0009-0004-0662-2078 (M. A. Mezentsev); 0009-0009-9385-4454(K.S. Kobernyk); 0000-0001-5293-5842 (G. G. Chetverykov)

2. Related Works

In the sweeping space of enormous information and database localization inside the Ukrainian scene, existing academic talk discloses a bunch of features relevant to the challenges and imminent resolutions. Understanding the history and latest techniques of big data and database management is crucial for navigating database localization in Ukraine effectively. In a general sense, database administration in Ukraine requires information localization techniques custom fitted to suit:

- Etymological characteristics,
- Social subtleties
- Administrative systems interesting to the locale.

While database design standards remain consistent across different regions, localizing databases for Ukraine requires a nuanced approach to preserve the relevance and accuracy of information within its specific context [3].

The insightful body of work shows a growing awareness of the important role of database localization in the Ukrainian big data scene. Researchers and specialists alike have set out on a travel of investigation, sending assorted techniques and procedures to hook with the unmistakable challenges characteristic in relative database localization.

Among the preeminent contemplations is the adjustment of existing database administration frameworks (DBMSs) to meet the localization objectives of Ukrainian datasets. Vital considers, such as those referenced [4], scrutinize the degree to which conspicuous DBMSs—Oracle, MySQL, Microsoft SQL Server, and PostgreSQL — Provide support for the Ukrainian language while adhering to local regulations. Understanding the strengths and weaknesses of these database management systems (DBMSs) is essential for establishing effective localization techniques.

Additionally, the etymological complexities natural to database localization for Ukrainian datasets have captured insightful consideration. Talking about efforts, such as the previously mentioned references, exploring the linguistic details and grammatical nuances that affect Ukrainian database querying and information retrieval tools. By dismembering Ukrainian-specific etymological designs and semantic structures, researchers try to plan advanced questioning standards and information control strategies finely adjusted to the complexities of the neighborhood dialect.

At the same time, legal and regulatory compliance becomes a primary consideration in Ukrainian database localization [5]. Exacting laws overseeing data protection, security, and capacity command thorough adherence to endorsed strategies. Insightful examinations, as reflected in references and related writing [6], carefully examine the legal framework supporting data localization in Ukraine and assess its impact on database management techniques. A comprehensive understanding of the overly complex lawful and administrative landscape engages analysts to make localization systems that consistently adjust with Ukrainian authoritative orders.

Moreover, the talk amplifies to the catalytic part of innovative developments in impelling database localization moves forward. Cutting-edge advances, such as natural language processing (NLP) and machine learning, show promising roads for mechanizing the localization process. Investigate activities, as prove by the referenced studies, dive into NLP-driven techniques pointed at:

- Facts: Descriptions that define what contains the data and how they relate
- Constraints: Rules that limit actions allowed by the information system (IS) or on its users
- Action enablers: Guidelines that trigger some of the activities under specific conditions.
- Inferences: Rules used to deduce new facts from existing ones, often written as "if-then" statements
- Computations: Equations or algorithms that change numerical values into new ones
- Terms: Words, abbreviations, and phrases important to a business

By tackling the transformative potential of natural language processing and machine learning calculations, analysts looking forward to streamline the localization direction and expand the availability of Ukrainian enormous information storehouses.

In conclusion, academic research sheds light on the complex nature of database localization in the Ukrainian context. Researchers examine language, administrative, and technological aspects to create a comprehensive understanding of managing databases in Ukraine's growing big data ecosystem. Looking ahead, collaborative efforts and innovative strategies will be crucial in overcoming the challenges of database localization and unlocking Ukraine's vast potential in big data. In exploring related works and implementations concerning the localization of databases within the Ukrainian context, several notable examples emerge.

For example, projects like the Ukrainian National Language Corpus (UNLC) have been instrumental in understanding the linguistic intricacies relevant to database localization. The UNLC serves as a comprehensive repository of Ukrainian language data, aiding researchers and practitioners in developing more nuanced localization techniques tailored to Ukrainian etymological patterns. Furthermore, initiatives by Ukrainian academic institutions, such as the Kyiv School of Economics (KSE) and Lviv Polytechnic National University, have contributed significant research efforts to the field of database localization.

Their studies delve into the intersection of linguistic analysis, legal frameworks, and technological innovations, offering practical insights into overcoming the challenges inherent in adapting database systems to Ukrainian language and regulatory requirements. In the realm of industry applications, companies like Rozum Robotics and Grammarly have integrated sophisticated localization strategies into their software products. Rozum Robotics, for instance, employs advanced database localization techniques to enhance the functionality of their robotic systems for Ukrainian market demands.

Similarly, Grammarly leverages natural language processing algorithms to provide contextualized language suggestions and corrections tailored specifically for Ukrainian users.

3. Methods and Materials

During the research, choice of technologies for the development of a prototype database localization system was carefully considered and based on a deep analysis of current market needs and modern realities in the IT industry. Analysis allowed us to choose solutions that meet the highest standards of efficiency and availability for wide public of IT companies.

3.1. Methods and Technologies description

The proposed system prototype for the localization of large databases is focused on the challenges and opportunities of data translation from Russian to Ukrainian, integrating with usual and widespread database management technologies and most effective open source translation systems. In light of significant advances in the field of DBMS, from early developments such as IBM's System R and Oracle Database to modern solutions such as MySQL, PostgreSQL and MongoDB, our prototype architecture is aiming to get knowledge about the effective way of parsing and buffering data and the most effective way of processing such data. So the methods and technologies used may lose popularity or effectiveness but the main architecture can later be re-used using different technologies or aiming on different languages[7].

The main part of the system is the data retrieving module, which plays a key role in ensuring translation efficiency. During the research, three types of data retrieval modules were tested.

First type is called "Cache module" – this module puts retrieved data into cache memory until it reaches the maximum amount of words that specific translation system could handle in a single API request.

Cache module steps:

1. Extract data text entry
2. Put extracted data into array separated by special symbols "|"
3. Add special entry in array of ID's with data ID and number in previous array
4. Repeat steps 2 and 3 until number of words/letters in array matches maximum amount for single API call
5. Send completed array to translation service via Post API call
6. Collect response with translated data
7. Separate data with assigned ID by matching Array of ID's and translated array
8. Send data text into new DB instance

This type of data buffering is used to ensure high system performance by minimizing the amount of API calls made to translation services thus lowering the load on the network, but it can only be used if our dataset or database has single table entry no bigger than maximum that translation system will accept in single API call.

Second type is called “Divide module” – this module is designed to divide data entries into sections for later translation if the data entries are bigger than maximum allowed number of words in single API call. This module takes a large entry of text in database and puts it into array for later subtraction into parts, translation via API call and combining back together in another translated array.

Divide module steps:

1. Extract data text entry
2. Subtract defined maximum amount of words/letters for API call till first dot
3. Send the subtracted text to translation service for processing
4. Add translated text to final array
5. Send data text into new DB instance

Third type is called “Instant module” – this module is designed to test and compare impact on effectiveness while number of API calls are being sent for every entry of the database by sending instant call for translation after receiving extracted data.

Instant module steps:

1. Extract data text entry
2. Send text to translation service for processing
3. Receive translated text
4. Send received text into new DB instance

Easy integration with different types of DBMS provides a wide coverage of potential usage scenarios for this three modules. The prototype is designed to support easy integration with relational and non-relational databases, giving engineers the freedom to choose the technology that best suits their needs.

This research doesn’t include custom offers from services shown on Figure 1 where they can expand number of characters per call, remove some of the restrictions or add any other additional functionality to your interaction with translation service. Lack of such data about custom offers – won’t affect our study about architecture efficiency and such additions may only help engineers in creating experienced prototype of such system by raising its efficiency.

The most important step in this research is collecting statistics about mistakes that can be made by services used during experiment with our test data[8]. In order to define the percentage of logical or regular mistakes made by every translation service the tables – translated test data were checked and the current service counter is updated in case of mistakes found. The percentage of mistakes is counted by finding mistake percentage in average count of translated and non-translated words (1)

$$d/\left(\frac{(b+c)}{2} \times 0{,}01\right) = a \qquad (1)$$

where b is a number of not translated words, c is a number of translated words and d is a number of found mistakes during translation so a represents our percentage of mistakes made by a service after translating b amount of words.

After gathering statistics on translation mistakes and assessing the architecture efficiency, it is crucial to consider the implications of these findings. Understanding the patterns of mistakes and the impact of custom offers on translation services can provide valuable insights for both researchers and engineers [9].

To demonstrate the capabilities of the prototype, a dataset with data in the Russian language was selected. This choice is not accidental, as it allows you to test the system in conditions that are as close as possible to the real needs of the Ukrainian market. Translation from Russian into Ukrainian requires not only high accuracy and preservation of context, but also a deep understanding of cultural and linguistic nuances, which poses complex tasks to the system, the solution of which requires the application of the latest achievements in the field of machine learning and natural language processing in different translation services used in research.

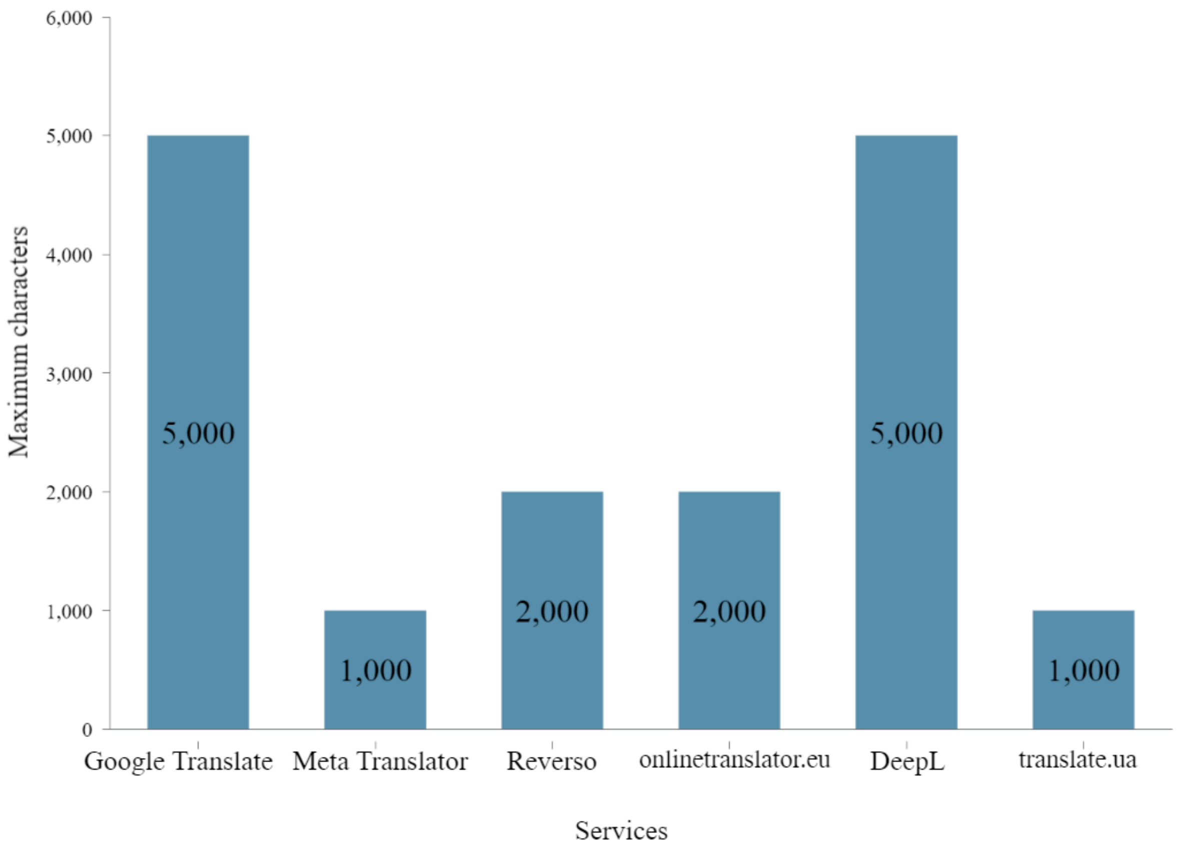

Figure 1: Chart of different translation services Max. characters per 1 call

The dataset that was uploaded into database contains around 140 000 rows with dependencies and 3 tables for short texts. Localized example can be seen in Table 1. And about 70 000 rows for long paragraphs in single table. Localized example can be seen in Table 2.

That kind of dataset was used to test English translation accuracy and can be found in open source by the name ”RuBQ” now we will use it to test localization from russian to Ukrainian.

Table 1
Data entry example for short texts

Uid	Question	Answer
2	"Хто автор п'єси «Ромео і Джульєта» ?"	"Шекспір"
3	" Як називається столиця Румунії?"	"Бухарест"

The “Uid” column is used to represent unique id that is assigned to a specific column and can be used to connect relative collumns from different tables. In our case we made a relation between “Question” table and “Answer” table.

The “Question” column is representing text field that contains questions for a different topics. Such type of data has to be translated correctly so it will be understood after translation – that factor can be an indicator for logical mistakes made by translation services.

The “Answer” column is basically the same as “Question” by type. It contains answers for questions stored in previous column. It also has to be properly translated because if it is not – the whole row in database becomes invalid and unusable. Visualization of such database relations with are shown on Figure 2.

Table 2
Data entry example for long texts

Uid	Paragraph
3098	Впродовж британського правління аж до отримання незалежності від Малайзії в 1965 році Сінгапур і Республіка Китай мали дипломатичні відношення, які продовжилися і після проголошення незалежності.

The “Paragraph” column is representing some historical data that is stored under specific id. The size of the paragraph can vary and they can be much bigger, up to 1000 symbols.

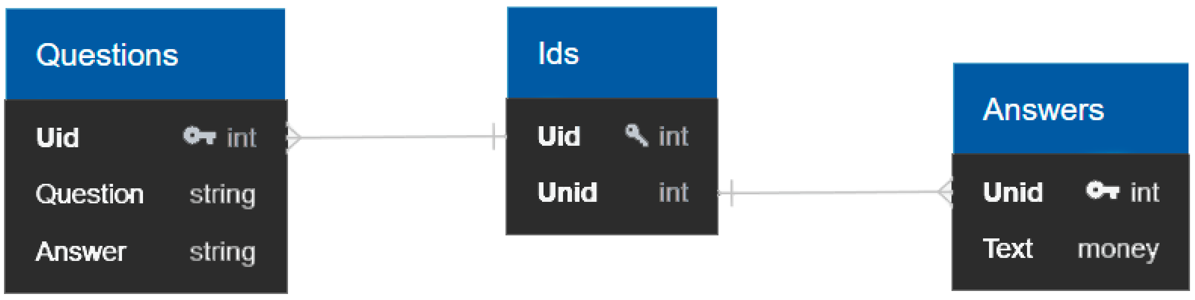

Figure 2: First dataset database visualization

Visualized database is MySQL instance that is used during tests and contains three databases. Databases has many-to-many relations using separate table to store Ids in between the connection.

4. Experiment

The series of experiments were conducted and aimed at understanding the best database management system for accommodating Ukrainian language-specific data.

The experiment will involve acquiring a dataset containing JSON data representative of language content needing translation. This dataset was imported into MySQL database, to measure time needed to parse large quantities of data in combination with time spent on translation and data transfer to translation services.

Key metrics for evaluation will include percentage of translation mistakes, single row translation time, average time needed to perform full translation and data-storing cycle. Through comprehensive experimentation and analysis, we aim to identify the architecture that best addresses the bigdata localization challenges in Ukrainian big data[10].

By understanding the strengths and limitations of different database solutions, database localization strategies can be addressed and contributed to the development of more efficient and effective data management practices tailored to Ukrainian language requirements.

Steps that were performed during experiment:

1. JSON data from open source
2. Loaded data into MySQL cluster
3. Filtered text from the data
4. Sent the chosen text to translator
5. Got data back from translator and saved it into new instance

Visualized version of the steps did during the experiment is shown on the figure 3.

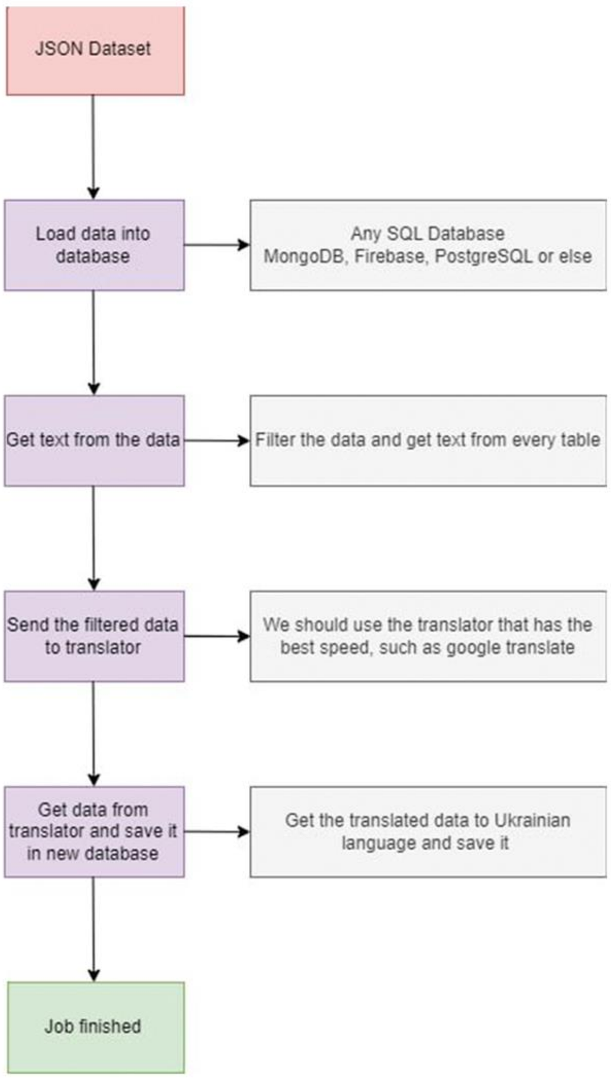
JSON Dataset
Load data into database
Any SQL Database MongoDB, Firebase, PostgreSQL or else
Get text from the data
Filter the data and get text from every table
Send the filtered data to translator
We should use the translator that has the best speed, such as google translate
Get data from translator and save it in new database
Get the translated data to Ukrainian language and save it
Job finished

Figure 3: Vizualization of steps during the experiment

While choosing a database for the experiment, we should look at various parameters to find the one we need. Figure 4 shows the comparison of the most popular SQL databases. The choice of MySQL determined for several reasons:

- MySQL is the most popular database that is widely used it the software engineering world.
- MySQL is one of the best choices for performance and stability. It can handle large amounts of data of any kind.
- provides quick access to the data
- Supports different types of queries, flexible data control
- Has convenient data management

Overall, it is the best choice for our goals, since we have to load a large amount of data and mySQL allows us to have easy control over the data and conduct any manipulations we need.

Table 3: databases comparison

Parameter	MongoDB	Firebase	MySQL
Ukrainian Language Support	Yes	Yes	Yes
Database Type	NoSQL	NoSQL	Relational
Performance	Average	Average	High

Scalability	Average	High	High
Data Management	Limited	Convenient	Convenient
JSON Support	Yes	Yes	Yes
Community Support	Large	Large	Large
Security Features	Limited	Comprehensive	Comprehensive
Indexing Capabilities	Limited	Limited	Extensive
Replication and Clustering	Limited	Limited	Comprehensive
Data Integrity	Limited	High	High
Stored Procedure Support	No	No	Yes

5. Results

Evaluated results represent 2 different types of data tested and 6 different services used in MySQL database. Results were split by subchapters for every service used in experiment with 2 tables containing short data and long data.

5.1. Google translate

Table 4 shows mistakes percentage, time used for single row translation of short texts, average process time from parsing data to translating and finally uploading data to database.

Table 4
Results for Google service with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~0.720 ms	~0.187 ms	854	5.76%

Table 5 shows time used for long texts.

Table 5
Results for Google service with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~0.419 ms	~0.183 ms	18969	5.61%

5.2. Meta translator

Table 6 shows time used for short texts.

Table 6
Results for Meta translator with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~2.11s	~0.620ms	4270	4.12%

Table 7 shows time used for long texts.

Table 7
Results for Meta translator with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~1.56s	~0.578ms	94 845	3.72%

5.3. Reverso

Table 8 shows time used for short texts.

Table 8
Results for Reverso context service with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~1.19s	~0.691ms	2135	4.16%

Table 9 shows time used for long texts.

Table 9
Results for Reverso context service with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~0.998ms	~0.648ms	47423	3.97%

5.4. Onlinetranslator.eu

Table 10 shows time used for short texts.

Table 10
Results for onlinetranslator.eu with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~2.55s	~0.791ms	2135	4.21%

Table 11 shows time used for long texts.

Table 11
Results for onlinetranslator.eu with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~1.99s	~0.723ms	47423	3.69%

5.5. DeepL

Table 12 shows time used for short texts.

Table 12
Results for DeepL service with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %

~8.72s	~4.94s	854	2.21%

Table 13 shows time used for long texts.

Table 13
Results for DeepL service with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~7.88s	~5.22s	18969	1.87%

5.6. Translate.ua

Table 12 shows time used for short texts.

Table 12
Results for translate.ua with short texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~5.92s	~2.32s	4270	0.89%

Table 13 shows time used for long texts.

Table 13
Results for translate.ua with long texts

Average process time	Time for translation of single row	API calls number	Mistakes %
~5.34s	~2.19s	94 845	0.72%

6. Discussion

After a deep analysis of recorded scores for process time listed in results chapter, it is clear that some of the services can perform translation faster than other ones but there's a dependency on speed versus quality because the percentage of mistakes is much higher if the "Google translator" is used and stays at (5.76%) for short texts and (5.61%) for longs texts. but at the same time it has the lowest process time of (~0.720 ms) for short texts and (~0.419 ms) for long texts and single row translation speed of (~0.184 ms) for both datasets on average.

Looking at the lowest mistakes rate from "translate.ua" with a (0.89%) of mistakes for short texts and (0.72%) for long texts dataset – it has much better translation accuracy thanks to the fact that it's local Ukrainian translator and it is more specific to our needs. But on the other hand, there is need to make much more API calls, due to limitation of 1000 words. Because of that, average process time is highly increased up to 5.92 seconds for short texts and 5.34 seconds for the long ones. The process time depends on the several reasons, including speed of ethernet connection, computer hardware capabilities and the service servers.

Such popular services from big corporations, as google translate or Meta translator has the best hardware and servers and for that reason the API has faster response rate. Collected data were split by short text average process time due to highest mistakes percentage, as services use less context analyzation in short texts. More detailed line chart is shown on the figure 4.

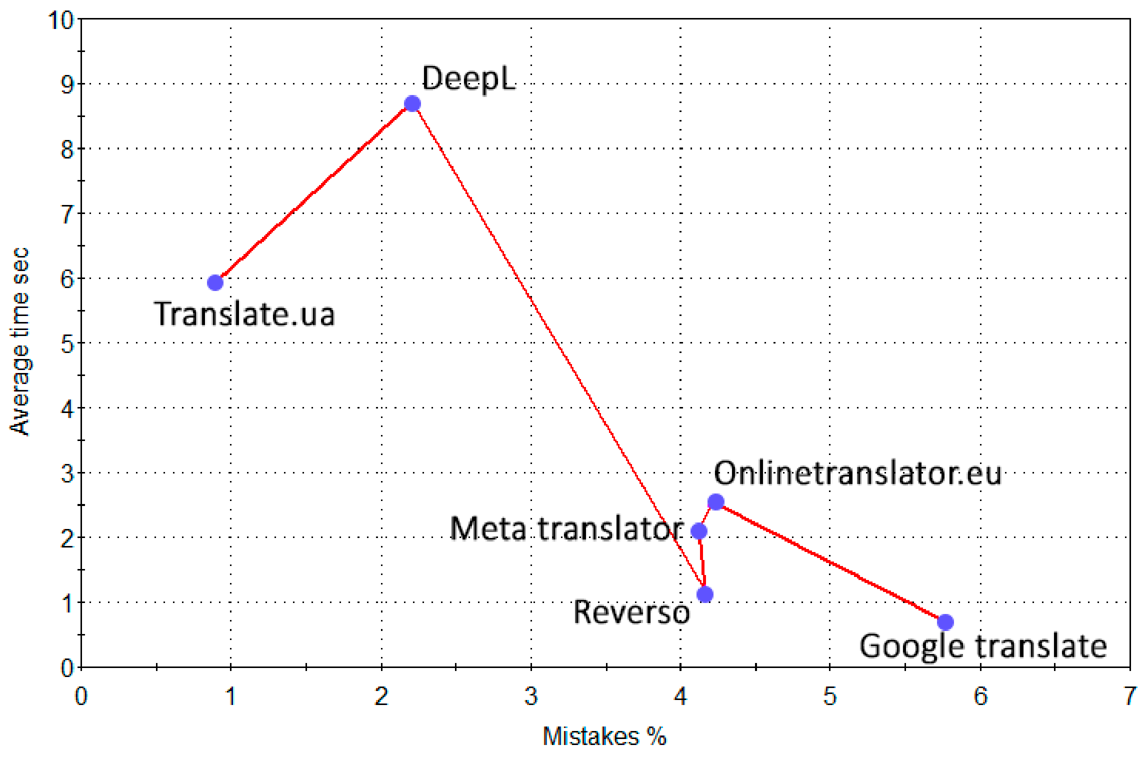

Figure 4: Visualization of Average time / Mistakes chart

Based on achieved results, each translator has it's own benefits and choice of used service depends on the goal we want to achieve. If the crucial part is API call speed, the suitable service should have good servers with low response time and high symbol restriction as for "Google Translate" or "Meta". On the other hand, the best choice for translation accuracy is local translator, since they are more adapted to the language environment and can show low mistakes percentage even with small amount of context for translated texts.

As a matter of fact the decision of Database and it's workload has minimal impact on average row process time as we used locally hosted solution, but if database location, server load and internet connection is applied – time can be increaced significantly.

7. Conclusion

Upon discussing the Big Data field in Ukraine and the difficulties associated with localizing databases, several important topics come to light.

First off, there are special reasoning localizing databases in Ukraine. By respecting national interests and guaranteeing that sensitive information stays under national control, localizing data can improve data sovereignty, security, and regulatory compliance. However, putting into practice successful database localization techniques - require a strong technological foundation, technological investment, and adherence to global data standards.

Secondly, there is special connection between the localization of databases in Ukraine and more general problems with data privacy and data availability.

Thirdly, Adherence to localization specifications could result in extra expenses and regulatory complications, which could affect corporate operations and innovation. Furthermore, data localization strategies may obstruct international collaboration and cross-border data flows, which would reduce prospects for global integration and economic growth.

In summary, the resolution of database localization issues in Ukraine necessitates a sophisticated strategy that strikes a compromise between national and international concerns. Ukraine can successfully traverse the challenges of data localization and realize the full potential of its big data ecosystem for the good of society at large by adhering to the values of awailability, accountability, and inclusion.

8. References

[1] A. Kopp, D. Orlovskyi, S. Orekhov, An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure, CEUR Workshop Proceedings 2870 (2021) 1274-1291.

[2] M. Garcarz, Legal Language Translation: Theory behind the Practice, CEUR Workshop Proceedings, Vol-3171 (2022) 2-2.

[3] I. Shubin, A. Kozyriev, V. Liashik, G. Chetverykov, Methods of adaptive knowledge testing based on the theory of logical networks, in: Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems, COLINS 2021, Lviv, Ukraine, 2021, pp. 1184– 1193.

[4] Bur, M., & Stirewalt, K. (2022). ORM ontologies with executable derivation rules to support semantic search in large-scale data applications, Proceedings - ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems, MODELS 2022: Companion Proceedings, p. 81.

[5] A. Kopp, D. Orlovskyi, S. Orekhov, An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure, CEUR Workshop Proceedings 2870 (2021) 1274-1291.

[6] Kulik, A., Chukhray, A., & Havrylenko, O. (2022). Information Technology for Creating Intelligent Computer Programs for Training in Algorithmic Tasks. Part 1: Mathematical Foundations. System Research and Information Technologies, 2022(4), 27-41. doi:10.20535/SRIT.2308-8893.2021.4.02

[7] H. Falatiuk, M. Shirokopetleva, Z. Dudar, Investigation of architecture and technology stack for e-archive system, in: 2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 – Proceedings, p. 229.

[8] K. Herud, J Baumeister, Testing Product Configuration Knowledge Bases Declaratively, in: LWDA 2022 - Workshops: Special Interest Group on Knowledge Management (FGWM), Knowledge Discovery, Data Mining, and Machine Learning (FGKD) and Special Interest Group on Database Systems (FGDB), CEUR Workshop Proceedings, vol. 3341, pp. 173-186.

[9] Igor Shubin, Andrii Kozyriev, Method for Solving Quantifier Linear Equations for Formation of Optimal Queries to Databases in: Computational Linguistics and Intelligent Systems 2023, Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems. vol. 449-459

[10] Wu Aiyan, Zhang Yongmei, Yang Shang. (2022). A Method for Scientific Cultivation Analysis Based on Knowledge Graphs, in: 12th International Conference on Electronics, Communications and Networks, CECNet 2022.