DOI: https://doi.org/10.15276/aait.06.2023.22
UDC 004.91

Data preprocessing and tokenization techniques for technical Ukrainian texts

Sergii V. Mashtalir[1)]
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus Author ID: 36183980100
Oleksandr V. Nikolenko[2)]
ORCID: https://orcid.org/0000-0002-6422-7824; oleksandr.nikolenko@gmail.com
[1)] Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine
[2)] Uzhhorod National University, 14, University Str. Uzhhorod, 88000, Ukraine

ABSTRACT

The field of Natural Language Processing (NLP) has witnessed significant advancements fueled by machine learning, deep learning, and artificial intelligence, expanding its applicability and enhancing human-computer interactions. However, NLP systems grapple with issues related to incomplete and error-laden data, potentially leading to biased model outputs. Specialized technical domains pose additional challenges, demanding domain-specific fine-tuning and custom lexicons. Moreover, many languages lack comprehensive NLP support, hindering accessibility. In this context, we explore novel NLP data preprocessing and tokenization techniques tailored for technical Ukrainian texts. We address a dataset comprising automotive repair labor entity names, known for errors and domain-specific terms, often in a blend of Ukrainian and Russian. Our goal is to classify these entities accurately, requiring comprehensive data cleaning, preprocessing and tokenization. Our approach modifies classical NLP preprocessing, incorporating language detection, specific Cyrillic character recognition, compounded word disassembly, and abbreviation handling. Text line normalization standardizes characters, punctuation, and abbreviations, improving consistency. Stopwords are curated to enhance classification relevance. Translation of Russian to Ukrainian leverages detailed classifiers, resulting in a correspondence dictionary. Tokenization addresses concatenated tokens, spelling errors, common prefixes in compound words and abbreviations. Lemmatization, crucial in languages like Ukrainian and Russian, builds dictionaries mapping word forms to lemmas, with a focus on noun cases. The results yield a robust token dictionary suitable for various NLP tasks, enhancing the accuracy and reliability of applications, particularly in technical Ukrainian contexts. This research contributes to the evolving landscape of NLP data preprocessing and tokenization, offering valuable insights for handling domain-specific languages.

Keywords: Multilingual natural language processing; data preprocessing; tokenization; technical Ukrainian texts; lemmatization

***For citation*: Mashtalir S. V., Nikolenko O. V. “Data preprocessing and tokenization techniques for technical Ukrainian texts”. *Applied Aspects of Information Technology.* 2023; Vol. 6 No. 3: 318–326. DOI: https://doi.org/10.15276/aait.06.2023.22**

INTRODUCTION AND LITERATURE REVIEW

Text mining has long been one of the priority areas for the development of scientific research. This is due to the fact that the creation of a compact text description of a particular physical process or technical problem is one of the most intuitive solutions. At the same time, the science development in general and the existence of appropriate tools for deep neural networks also make it possible to big data text analysis, which is typical for a number of tasks. In particular, it can be noted that such areas as medical diagnostics, when marking algorithms can be used to big data text analysis [1], there are also approaches that allow you to extract the necessary information under various kinds of noise conditions [2].

Another direction in the big data text analysis is various kinds of abstracting or summarization, which allows you to make a brief description that can be used for various tasks related to the data search systems [3, 4], [5, 6]. At the same time, in most cases, the extraction of textual information is reduced either to the problem of data classification [7], or various data clustering [8, 9]. However, it should also be noted that most of the works related to the extraction of text data do not take into account the specifics of a particular language, but are a common toolkit.

In this regard, the direction of text analysis, which is associated with taking into account the features of various kinds for a particular language, is quite promising. These developments have not only enabled more natural and intuitive human-computer interactions but have also expanded the scope of NLP applications.

© Mashtalir S., Nikolenko O., 2023

This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/deed.uk)

The field of Natural Language Processing (NLP) has undergone a profound transformation in recent years, driven by advancements in machine learning, deep learning, and artificial intelligence.

Despite the wealth of data available, NLP systems still grapple with issues related to data incompleteness and errors [10, 11]. Inaccurate or incomplete training data can lead to biased or erroneous model outputs, affecting the reliability of NLP applications. Addressing these issues requires the development of more robust data curation and cleaning processes.

Natural language processing faces challenges when dealing with highly specialized scientific or technical domains that employ domain-specific terminology. Adapting NLP models to accurately understand and generate content in these domains remains an ongoing challenge, necessitating domain-specific fine-tuning and custom lexicons.

The majority of NLP vocabularies and libraries are well-developed for the Top-10 languages, leaving many languages with limited or no NLP support [12, 13]. This imbalance restricts the accessibility and utility of NLP technologies for speakers of less-represented languages.

While NLP models have achieved impressive results in monolingual environments, they encounter difficulties in multilingual scenarios [14]. Languages that share numerous common words and linguistic roots pose challenges in disambiguation and accurate language processing. The development of effective multilingual NLP models remains an area of active research.

The problem this article is about at hand revolves around the utilization of a dataset comprising names of automotive repair labor entities. These data have been extracted from the Garage Management System and are notable for their propensity to harbor errors, technical abbreviations, and domain-specific terminology. Furthermore, the dataset encompasses predominantly Ukrainian and Russian languages, occasionally exhibiting a blend of the two, commonly referred to as "Surzhik". Additionally, we possess annotated data in the form of a detailed classifier of automotive service jobs in both Ukrainian and Russian languages.

The overarching objective entails the classification of each labor entity into predefined categories. To accomplish this, the initial steps involve a comprehensive data cleansing and preprocessing pipeline, including tokenization see Fig. 1 and [15]. The aim is to transform the raw natural language text into a binary vocabulary conducive to subsequent classification tasks.

To mitigate the substantial impact of vocabulary size on computational resources and processing time [14], it is imperative to implement vocabulary normalization as an integral part of its inception.

THE GOAL OF PAPER

The goal of this paper is to create a normalized text data model for bilingual technical text data sets in Ukrainian and Russian in the presence of errors and incomplete data.

PROPOSED APPROACH

Classical NLP preprocessing pipeline contains some preliminary (sentence segmentation, word tokenization), frequent (stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing) and other steps (normalization, language detection, code mixing, transliteration) [15, 16].

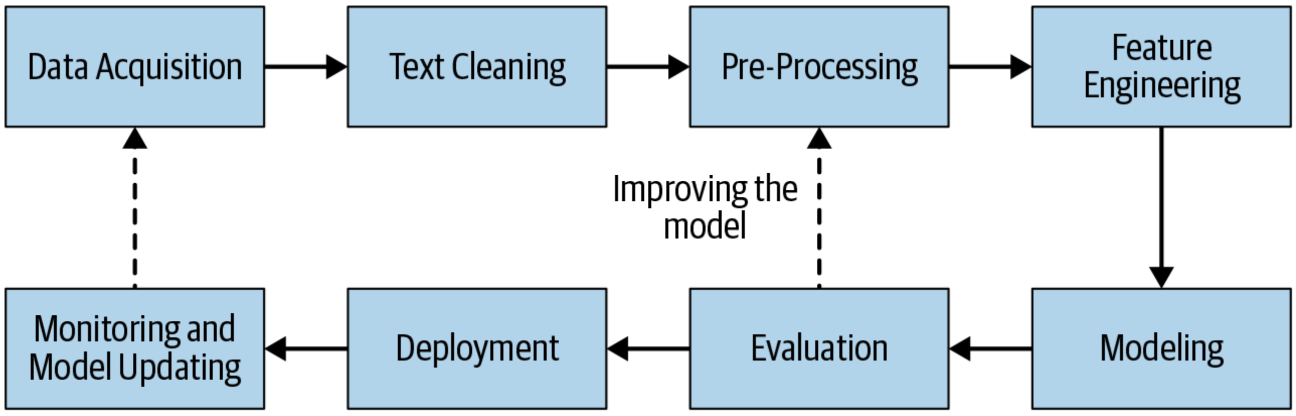

Fig. 1. **Natural Language Processing Pipeline**
***Source:* compiled by the [15]**

In our data preprocessing pipeline, certain modifications were necessitated due to the origin of the data. Specifically, some conventional steps such as sentence segmentation were deemed unnecessary. Conversely, we introduced additional procedures tailored to the dataset's characteristics, which included language detection, specific Cyrillic characters approach, the disassembling of compounded words and handling of specific shortcuts and abbreviations.

1. Language identification

Language identification is accomplished through the utilization of two distinct approaches:

1. **Identification of Specific Characters**: This approach involves the recognition of language-specific characters, such as "і," "ї," "є," and "ґ" for Ukrainian, as well as "ы," "ъ," "э," and "ё" for Russian.

2. **Word Counting in Dictionaries**: A complementary method relies on counting the occurrences of words found in both Ukrainian and Russian dictionaries, as detailed in the “Lemmatization” chapter.

Following the language identification process, all data undergo translation into Ukrainian. This translation is facilitated by two custom correspondence dictionaries, which are elaborated upon in the chapters titled “Translation of Tokens from Russian to Ukrainian” and “Synonyms”.

2. Text lines normalization

Text line normalization encompassed a series of standardization procedures. These encompassed segregating numbers and punctuation symbols with spaces, uniformly converting all characters to lowercase, substituting backslashes (“\”) with regular forward slashes (“/”), and replacing underscores (“_”) with spaces. Additionally, our specific task demanded the normalization of various types of apostrophes into a singular format. This process also extended to the treatment of certain Cyrillic characters, such as transforming “ґ” to “г” and "ё" to "е". Furthermore, prior to the removal of stop words and special characters, common abbreviations featuring slashes or hyphens were substituted with their expanded counterparts, a feature particularly relevant to the context of garage repair texts (e.g., "к-т" denoting "комплект" (kit), "д/м" signifying "демонтаж / монтаж" (mounting / dismounting) and "о/р" representing "охолоджуюча рідина" (cooling fluid) among others). Subsequently, all special characters, with the exception of the apostrophe, which holds linguistic significance in the Ukrainian language, and designated stop words were eliminated from the text.

3. Stopwords

Our compilation of stopwords comprised a comprehensive set, encompassing both Ukrainian and Russian languages. Notably, certain stopwords present in the general set were excluded, given their relevance to our classification task. For instance, "ТО," an abbreviation for "технічне обслуговування" (technical maintenance), and "ніж," which could be interpreted as a noun (knife) and is pertinent to our directory, were retained. Conversely, additional stopwords were introduced that did not significantly contribute to the subsequent classification process. These included brand names of cars or parts, terms such as "auto," "automobile," "automotive," "service," and other highly generic words devoid of distinctive characteristics relevant to individual entities.

4. Translation of tokens from Russian to Ukrainian

Since we possess detailed classifiers containing Ukrainian and Russian versions of names, and since most names share an identical number and order of words in both versions, we were able to automatically construct a correspondence dictionary between Russian and Ukrainian words.

For each name in the dictionary, we iteratively compare Ukrainian and Russian name versions.

1. Split both versions into token lists.

2. If the lengths of the token lists are equal, iterate through the tokens and increment a “counter” by 1 for each corresponding Ukrainian-Russian token pair. If the number of tokens in the names is not the same, add the missing tokens to the “omitted” category, which is then manually checked. We also skip tokens that are identical in both versions.

3. We obtain correspondence dictionaries where each Russian token corresponds to Ukrainian tokens that were found in the same position in the sentence, along with the number of such occurrences. We select the translation option that occurred most frequently as the most likely correct one. Consequently, we have a dictionary where each Russian token can be matched to its Ukrainian equivalent.

4. Additional steps included manual verification of token translations that differed significantly according to the Jaro-Winkler metric [16], as well as the addition of translations for omitted names with differing token counts.

5. Tokenization of “concatenated” tokens (with missing spaces)

The algorithm takes as input a unique set of tokens derived from the data we intend to classify subsequently. It searches for concatenated tokens within this set.

For each input token:

1. Check whether it starts or ends with a token known to us.
2. If so, separate it and add it to the "parts" list.
3. Repeat steps 1-2 until we traverse the sorted list of known tokens.
4. Anything that remains unprocessed is added to the list of parts. Remove all parts that are absent from the set of known tokens.
5. If, after this process, more than one part is obtained, identify it as a concatenated token. Add both the original concatenated token and its decomposition to a “dictionary” of token decompositions, which is used later to replace such tokens in input strings.

Table 1 provides illustrative instances of concatenated tokens prior to and token separation following the application of the algorithm.

This token concatenation search is relatively basic in nature, as it can only break tokens that start or end with reference tokens without typographical errors. In the worst-case scenario, a token may not only be concatenated but also contain errors within its constituent parts, in which case the algorithm will fail to identify it. However, concatenated tokens themselves are relatively infrequent, and concatenated tokens with errors are even rarer. Developing a more complex algorithm to address such cases would entail significant computational costs. Therefore, we have chosen to implement this straightforward approach.

Table 1. **Concatenated tokens examples**

Concatenated tokens	Separated tokens
бампераспереднего	бампера переднего
Замінаправого	заміна правого
Супортасупорта	супорта
повітрянозаборного	повітря заборного
Блокступиці	блок ступиці

Source: **compiled by the authors**

6. Spelling correction

Spelling errors are identified within tokens that do not exist in reference dictionaries; otherwise, they are considered correct and skipped. Essentially, this process entails the search for the most similar words among those present in the token sets from reference dictionaries, including all their inflected forms found in Ukrainian and Russian noun and adjective dictionaries.

For input, we receive a unique set of tokens from the data that we plan to classify subsequently. Typographical errors are sought within this set. The error detection occurs in two stages.

1. In the first stage, **search for the nearest match for tokens** that differ by no more than 2 characters from existing tokens. The match is found using the Jaro-Winkler [17] similarity function and is accepted if the similarity value exceeds a predefined threshold. The Jaro-Winkler metric assigns greater weight to token "prefixes" (letters at the beginning of the token).

2. In the second stage, the match is also sought using the same function but without a limit on the maximum of 2 character differences, hence, the threshold value is higher compared to the first stage. Therefore, in the first stage, we tolerate lower “similarity”, as long as the token differs from existing ones by no more than 2 characters, while in the second stage, greater dissimilarity is allowed, but the similarity value requirement is higher.

In essence, during the first stage, the distance function allows for smaller “similarity”, but tokens must differ from existing ones by no more than 2 characters. In the second stage, greater differences (more than 2 characters) are allowed, but the similarity requirement is stricter.

7. Motivation for choosing the jaro-winkler metric

The Jaro-Winkler Measure is a measure of *similarity* / *distance* between two text sequences. It uses the prefix scaling factor *p*, which provides a higher score to sequences that match at the beginning up to a specified prefix length *l*. The higher the Jaro-Winkler similarity measure, the more similar the two text sequences are. The score is normalized such that 0 indicates no similarity, and 1 indicates a perfect match. *Similarity* and *distance* are inversely related, and their correspondence is established by the formula *distance* = 1 - *similarity*. The Jaro-Winkler Measure is a modification of the Jaro measure.

The Jaro similarity sim_j of two text strings s_1 and s_2 is determined by the following formula:

$$sim_j = \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & if\ m > 0 \end{cases},$$

where $|s_i|$ is length of string s_i; *m* is number of matching characters; *t* is number of transpositions.

Two characters from s_1 and s_2 are considered a match if they are identical and located no more than *p* positions apart. If no matches are found, the algorithm stops and returns a *similarity* score of 0. If matches are found, the number of transpositions is then calculated. A transposition occurs when a corresponding (matching) character is not in its correct position, and the number of corresponding characters not in their correct position, divided by 2, yields the number of transpositions.

The Jaro-Winkler similarity sim_{jw} of two text strings s_1 and s_2 is determined by the following formula:

$$sim_{jw} = sim_j + lp\,(1 - sim_j),$$

where sim_j is Jaro similarity of text strings s_1 and s_2; *l* is length of the common prefix at the beginning of the string, maximum of 4 characters; *p* is a constant scaling coefficient that adjusts the estimate in the direction of increasing values in the presence of a common prefix: the standard value is 0.1;

or

$$sim_j = \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) + lp\left(1 - \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)\right) & if\ m > 0 \end{cases}.$$

There are many distance metrics, among others Levenshtein Distance, Indel (Insertion-Deletion) Distance and Hamming Distance.

Levenshtein Distance. The minimum number of single-character operations (insertions, deletions, and substitutions) required to transform one text string into another [18]:

- *Insertion*: автообіль → авто**м**обіль
- *Deletion*: автом**и**обіль → автомобіль
- *Substituion*: автомо**ь**іль → автомо**б**іль

According to Levenshtein pair of редуктор → редуктор**ний** has a distance of 3 (3 insertions) and pair of ремк**ом**плект → ремк**мо**плект has a distance of 2 (2 substitutions).

Indel (Insertion-Deletion) distance. The minimum number of insertions and deletions of characters required to transform one text string into another. Substitutions are not allowed, but each substitution can be accomplished by a pair of a deletion and an insertion, making this distance equivalent to the Levenshtein distance with a substitution weight of 2.

Hamming distance. The number of positions where two strings of equal length differs. It represents the minimum number of substitutions required to transform one string into another and can only be applied to sequences of equal length.

Jaro-Winkler Metric is more complex than simple distance algorithms based on counting basic character operations. It provides a real value between 0 and 1, making the distance values more informative and suitable for comparison and sorting. Additionally, it gives more significance to prefixes, which is a useful property when dealing with "typo searching". Spelling correction is used not for spelling errors only but as well for words with variations in their endings, sharing a common prefix. Giving more weight to prefixes increases the chances of correctly identifying such "typos".

Differences in short sequences are more significant than in long sequences. Therefore, the metric's value must depend not only on the number of basic operations but also on the length of the sequences, allowing for more accurate word comparisons.

Table 2 shows comparative example using the Levenshtein distance and Jaro-Winkler metric, with short and long words.

Table 2. **Levenshtein distance and Jaro-Winkler Metric Comparison**

Comparable strings	Leven-shtein Distance	Jaro-Winkler Metric
кожух – кожуха	1	0.966
сайлентблок – сайлентблока	1	0.983
ко**жух** – ко**леса**	4	0.577
сайле**нт**блок – сайле**тн**блок**ів**	4	0.951

***Source:* compiled by the authors**

The Levenshtein metric erroneously yields equivalent distances for the latter two comparisons, which stands in contradiction to the substantial dissimilarity in the lengths of the respective sequences. In stark contrast, the Jaro-Winkler metric aptly delineates the similarities among pairs 1, 2, and 4 while appropriately highlighting the substantial dissimilarity within pair 3.

8. Lemmatization

Given that we are working with Ukrainian and Russian languages, which have a vast number of word forms, and the unavailability of as sophisticated NLP libraries as for English, we implemented lemmatization independently using electronic dictionaries.

The construction of dictionaries mapping word forms to their lemmas is performed by extracting data from electronic Ukrainian and Russian language dictionaries. During extraction, dictionaries of correspondences for specific word forms to certain lemma are created (for some word forms, multiple lemmas may exist). An inverse dictionary, mapping lemma to word forms, is also generated. Subsequently, for word forms with multiple corresponding lemmas, we choose a single lemma (usually the most frequently occurring one, or in the case of equal occurrences, the first in the list). Word forms corresponding to all other lemmas are attributed to the chosen lemma. Although this approach may result in minor drawbacks when one word form belongs to different parts of speech (e.g., adjectives and nouns in our case), such situations are rare. This approach is preferred to a scenario in which some inflections are lemmatized into one lemma while others into a different one.

After completing these steps, we establish a definitive dictionary of correspondences between noun cases and lemmas for use in preprocessing.

Additionally, during lemmatization, another correspondence dictionary mapping Russian lemmas to their Ukrainian counterparts is generated. To achieve this, we traverse the Russian-Ukrainian translation dictionary, searching for lemmas from the form-lemma dictionary described earlier for each pair of words. If lemmas are found for both words, and they are not identical, these lemmas are added to the lemma translation dictionary.

Table 3 presents a systematic exposition of the sequential evolution of the initial text as it undergoes the procedures of text cleansing, preprocessing, and lemmatization.

9. Separation of common prefixes in compound words

Prefixes such as "електро" (electro), "пневмо" (pneumo), "авто" (auto), and others, are separated from tokens to standardize different spellings (both combined and separate) of such words. This also enables the linkage of names containing suffixes of compound words written with and without such prefixes (e.g., "пневмонасос" (pneumopump) – which transforms to "пневмо насос" (pneumo pump) – and "насос" (pump) will be treated as similar tokens; otherwise, these two tokens would have been considered entirely different).

10. Deciphering abbreviations

We created a file containing common word abbreviations / acronyms and their corresponding expansions. Abbreviations (including those with slashes and hyphens) were manually identified from a large dataset of job names. During preprocessing, tokens representing abbreviations are replaced with their expanded versions as shown in the Table 4.

Table 3. **Text processing step by step**

Initial Data	**Before lemmatization (after normalization, spell-checking, disassembling concatenated tokens, stop-words, translation)**	**After lemmatization**
Заміна зливного трубопроводу гідропідсилювача рульового керування	заміна зливного трубопроводу гідро підсилювача рульового керування	заміна зливний трубопровід гідро підсилювач рульовий керування
Прозвонка проводки на розетку освещения 15 pin. фонари. на поворот от блока LCM. питания LCM. пневмосигнал	продзвонка проводки розетку освітлення 15 ліхтарі поворот блока lcm живлення lcm пневмосигнал	продзвонка проводка розетка освітлення 15 ліхтар поворот блок lcm живлення lcm пневмосигнал
Установка выхлопной системы (глушитель. катализатор. выхлопная труба. выхлопная гофра)	встановлення вихлопної системи глушник каталізатор вихлопна труба вихлопна гофра	встановлення вихлопний система глушник каталізатор вихлопний труба вихлопний гофр
Ремонт проводки на датчик давления масла. датчик давления картерных газов. масла. топлива	ремонт проводки датчик тиску масла датчик тиску картерних газів масла палива	ремонт проводка датчик тиск масло датчик тиск картерний газ масло паливо
Демонта/монтаж полурессоры с высверливанием направяющего штифта (стремянки сняты. третья ось правая сторона)	демонтаж монтаж напів ресори висвердлювання направляючий штифта стремянки знятий третья вісь правая бік	демонтаж монтаж напів ресора висвердлювання направляючий штифт стрем'янка знятий третій вісь правий бік

Source: **compiled by the authors**

Table 4. **Typical Abbreviations**

Abbreviation	**Complete text**
ШРУС	шарнір рівних кутових швидкостей
ГБО	газо балонне обладнання
ГПК	гідро підсилювач керма
АКБ	акумуляторна батарея
ECU	електронний блок керування

Source: **compiled by the authors**

11. Synonyms

This is the specific aspect of the Ukrainian language – existence of "Surzhyk" (blending of Ukrainian and Russian words) and the abundance of synonyms.

We compiled a file containing synonyms for words and word combinations. Synonyms were identified during data processing (including during the creation of translation dictionaries from a list of all encountered translations). Some synonyms were also added based on logical considerations.

All synonym words are unified into a single form.

12. Vocabulary size reduction

As previously noted, the final token vocabulary size plays a pivotal role in determining the computational time and machine learning burden incurred in subsequent stages of the pipeline. Consequently, every step we undertake should exhibit a significant reduction in the volume of tokens.

Table 5 illustrates the outcomes of the algorithm when applied to a dataset comprising 10.288 initial sentences. When commencing with an initial count of 6.062 unique tokens, a sequence of preprocessing steps – including normalization, spell-checking, token disassembly, removal of stop-words, translation, lemmatization, and more – results in a reduction to 2.484 tokens, signifying a remarkable 60 % decrease in the original vocabulary size. It is worth noting that since computational time exhibits exponential growth in relation to vocabulary size, this 60 % reduction in vocabulary size translates to an impressive 84 % reduction in both computational time and computational load.

Table 5. **Step-by-step Vocabulary Size Reduction**

Step	Stage	No. of Tokens
1	Initial number of unique tokens	6 062
2	After normalization	4 847
3	Before lemmatization (after normalization, spell-checking, disassembling concatenated tokens, stop-words, translation)	3 991
4	After lemmatization	2 484

Source: **compiled by the authors**

RESULTS ACHIEVED

In conclusion, text mining has emerged as a pivotal domain in scientific research, driven by the need to succinctly describe complex physical processes and technical challenges. The advent of deep neural networks and the availability of substantial data have expanded the horizons of text analysis, finding applications in diverse fields such as medical diagnostics and data search systems.

However, the field of Natural Language Processing (NLP) faces several challenges. Data incompleteness and errors continue to plague NLP systems, necessitating robust data curation and cleaning. Moreover, specialized technical domains pose difficulties for NLP models due to domain-specific terminology, demanding domain-specific fine-tuning.

Language support in NLP remains skewed toward the Top-10 languages, limiting accessibility for speakers of less-represented languages. Multilingual scenarios pose additional complexities, especially for languages sharing common words and linguistic roots.

This article delves into the development of specialized technical vocabularies for bilingual datasets in Ukrainian and Russian, focusing on the automotive repair domain. The challenges addressed encompass data cleansing, language identification, normalization, spelling correction, lemmatization, and vocabulary reduction.

The use of the Jaro-Winkler metric in spelling correction is highlighted as a powerful tool for identifying similarities in text sequences. Its ability to consider prefixes and scale similarity values between 0 and 1 makes it well-suited for typo detection and spelling correction, especially in languages with complex word forms.

Additionally, the comprehensive preprocessing pipeline, including tokenization, language-specific character recognition, and synonym handling, contributes to a substantial reduction in vocabulary size. This reduction has significant implications for computational efficiency, resulting in an impressive 84 % reduction in computational time and load for the given dataset.

CONCLUSIONS

In conclusion, the model of technical text normalizations presented in this article offer valuable insights into addressing challenges in NLP, particularly in specialized bilingual domains, and underscore the importance of meticulous text preprocessing for efficient and accurate language processing.

Using the proposed approach, it was possible to reduce the terms vocabulary by 60 % (from 6062 to 2484 tokens), which, which gave to computational time and computational load 84 % reduction.

REFERENCES

1. Trivedi, H., Panahiazar, M., Liang, A., Lituiev, D., Chang, P., Sohn, J., Chen, Y., Franc, B., Joe, B. & Hadley, D. "Large scale semi-automated labeling of routine free-text clinical records for deep learning". *Journal of Digital Imaging*. 2018. [Scopus]. DOI: https://doi.org/10.1007/s10278-018-0105-8.

2. Nguyen, H. & Patrick, J. "Text mining in clinical domain: dealing with noise "*KDD '16: Proceedings of the 22nd ACM SIGKDD." International Conference on Knowledge Discovery and Data Mining*. August 2016. p. 549–558. DOI: https://doi.org/10.1145/2939672.2939720.

3. Gao, L. & Zhao, J. "Deep learning based network news text classification system" *MLMI '22: Proceedings of the 2022 5th International Conference on Machine Learning and Machine Intelligence*. September 2022. p. 52–57. [Scopus]. DOI: https://doi.org/10.1145/3568199.3568207.

4. Yang, Z., Zhenghan, C. & Hua, Z. "Automatic extraction of paper research methods based on multi-strategy". *ICCIP '19: Proceedings of the 5th International Conference on Communication and Information Processing*. November 2019. p. 196–200. [Scopus]. DOI: https://doi.org/10.1145/3369985.3370025.

5. Sulova, S. "Text mining approach for identifying research trends". *CompSysTech '21: Proceedings of the 22nd International Conference on Computer Systems and Technologies*. 2021. p. 93–98. [Scopus]. DOI: https://doi.org/10.1145/3472410.3472433.

6. Fisher, M., Albakour, D, Kruschwitz, U. & Martinez, M. Recognising Summary Articles. *Advances in Information Retrieval*. 2019. p.69–85. DOI: https://doi.org/10.1007/978-3-030-15712-85.

7. Xylogiannopoulos, K., Karampelas, P. & Alhajj R. "Text mining for malware classification using multivariate all repeated patterns detection". *ASONAM '19: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. August 2019. p. 887–894. [Scopus]. DOI: https://doi.org/10.1145/3341161.3350841.

8. Lachana, Z, Loutsaris, M.A. & Charalabidis, Y. "Clustering legal artifacts using text mining". *ICEGOV '21: Proceedings of the 14th International Conference on Theory and Practice of Electronic Governance*. 2021. p. 65–70. DOI: https://doi.org/10.1145/3494193.3494202.

9. Ren, S. "Multi-media content clustering and computer intelligent analysis by text mining". *AIAM2021: 2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture*. October, 2021, p. 2276–2285. [Scopus]. DOI: https://doi.org/10.1145/3495018.3501088.

10. Arenas, M., Botoeva, E., Kostylev, E. & Ryzhikov, V. "A note on computing certain answers to queries over incomplete databases". *In: CEUR Workshop Proceedings. Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web.* Montevideo: Uruguay. 2017.

11. Li, Y., Currim, F. & Ram, S. "Data completeness and complex semantics in conceptual modeling". *The Need for a Disaggregation Construct. Journal of Data and Information Quality*. 2022; 14 (4), Article No. 22: 1–21. [Scopus]. DOI: https://doi.org/10.1145/3532784.

12. Sebastian, M. P., & G, S. K. "Malayalam natural language processing: challenges in building a phrase-based statistical machine translation system". *ACM Transactions on Asian and Low-Resource Language Information Processing*. 2022; 22 (4), Article No. 117: 1–51. [Scopus]. DOI: https://doi.org/10.1145/3579163.

13. Butnaru, A.-M. "Machine learning applied in natural language processing". *ACM SIGIR Forum* . June 2020; 54 (1), Article No. 15: 1–3. DOI: https://doi.org/10.1145/3451964.3451979.

14. Vajjala, S., Majumder, B., Gupta, A. & Surana, H. "Practical natural language processing: A comprehensive guide to building real-world NLP systems". *Published by O'Reilly Media, Inc.* 2020.

15. Jurafsky, D. & Martin, J. "Speech and language processing". *Third Edition draft.* 2018. – Available from: https://web.stanford.edu/~jurafsky/slp3/ed3book_jan72023.pdf. – [Accessed: 28 July, 2022].

16. Winkler, W. E. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage". *Proceedings of the Section on Survey Research Methods. American Statistical Association.* 1990. p. 354–359. – Available from: https://files.eric.ed.gov/fulltext/ED325505.pdf. – [Accessed: 28 July, 2022].

17. Cohen, W. W., Ravikumar, P. & Fienberg, S. E. "A comparison of string distance metrics for name-matching tasks (PDF)". *KDD Workshop on Data Cleaning and Object Consolidation*. 2003; 3: 73–78. – Available from: https://www.cs.cmu.edu/afs/cs/Web/People/wcohen/postscript/kdd-2003-match-ws.pdf. [Accessed: 28 July, 2022].

18. Levenshtein, V. I. "Binary codes capable of correcting deletions, insertions, and reversals". *Cybernetics and Control Theory*. 1966; 10 (8): 707–710.

Conflicts of Interest: The authors declares that there is no conflict of interest

Received: 18.07.2023
Received after revision: 08.09.20232
Accepted: 20.09.2023

DOI: https://doi.org/10.15276/aait.06.2023.22
УДК 004.91

Методи препроцесінгу та токенізації даних для технічних українських текстів

Машталір Сергій Володимирович[1)]
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus Author ID: 36183980100
Ніколенко Олександр Володимирович[2)]
ORCID: https://orcid.org/0000-0002-6422-7824; oleksandr.nikolenko@gmail.com
[1)] Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна
[2)] Ужгородській національний університет, вул. Університетська, 14. Ужгород, 88000, Україна

АНОТАЦІЯ

За останні роки галузь обробки природної мови (Natural Language Processing, NLP) пережила значні досягнення завдяки машинному та глибинному навчанню і штучному інтелекту, що розширило її застосування та покращило взаємодію між людиною та комп'ютером. Однак системи обробки природної мови стикаються з проблемами, пов'язаними з неповними та помилковими даними, що може призводити до побудови моделей з помилковими результатами. Спеціалізовані технічні області ставлять додаткові вимоги, вимагаючи налаштування моделей під конкретну галузь та використання власних специфічних термінів. Більше того, багато природніх мов не мають повноцінної підтримки в NLP. У цьому контексті ми досліджуємо нові методи попередньої обробки даних та токенізації, призначені для технічних українських текстів. Ми працюємо з набором даних, що містить назви операцій з галузі автомобільного ремонту, специфікою яких є наявність багатьох помилок та присутність специфічних термінів, часто у комбінації української та російської мов. Нашою метою є точна класифікація цих сутностей, що на першому етапі передбачає комплексну очистку даних, попередню обробку та токенізацію. Наш підхід модифікує класичну попередню обробку NLP, включаючи виявлення мови, розпізнавання конкретних кириличних символів, розклад складних слів на прості частини та обробку абревіатур. Нормалізація частин окремого речення стандартизує символи, видаляє розділові знаки та розшифровує абревіатури. Переклад з російської на українську мову здійснюється шляхом використання детальних довідників та автоматично створених словників відповідностей. Під час токенізації вирішуються питання злитих токенів, орфографічних помилок, спільних префіксів у складних словах та абревіатурах. Лематизація, особливо важлива для мов, які використовують відмінки, використовує великі наукові словники, які перетворюють словоформи у леми, з акцентом на називному відмінку іменників. В результаті створюється повний словник токенів, який може використовуватись у різних завданнях у сфері обробки природної мови. Повнота словника та унікальність окремих токенів підвищує точність та надійність їхнього застосування, особливо в технічних українських текстах. Це дослідження поглиблює існуючі методи і моделі попередньої обробки та токенізації даних в рамках NLP та надає конкретні інструменти для роботи з текстами із специфічних галузей.

Ключові слова: обробка природної мови; попередня обробка даних; токенізація; технічні українські тексти; визначення мови; лематизація; технічні словники

ABOUT THE AUTHORS

Sergii Volodymyrovych Mashtalir - Doctor of Engineering Science, professor. Professor of Informatics Department Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus: 36183980100
Research field: Image and video processing; data analysis

Машталір Сергій Володимирович - доктор технічних наук, професор. Професор кафедри Інформатики Харківського національного університету радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Oleksandr Volodymyrovych Nikolenko - Specialist on Applied Mathematics. PhD student at Uzhhorod National University, 14, University Str. Uzhhorod, 88000, Ukraine
ORCID: https://orcid.org/0000-0002-6422-7824: oleksandr.nikolenko@gmail.com
Research field: Natural language processing; big data; machine learning

Ніколенко Олександр Володимирович - спеціаліст по спеціальності «Прикладна математика». Здобувач ступеня доктора філософії у Державному вищому навчальному закладі «Ужгородський національний університет», вул. Університетська, 14. Ужгород, 88000, Україна