Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 4 (26)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

UDC 65.014.12
DOI: https://doi.org/10.30837/ITSSI.2023.26.033

M. GRINCHENKO, M. ROHOVYI

A MODEL FOR IDENTIFYING PROJECT SPRINT TASKS BASED ON THEIR DESCRIPTION

The subject of research in this article is the identification of project sprint tasks. **The purpose of the article** is to find approaches to reducing the risks of not fulfilling sprint tasks. The article solves the following **tasks**: analyzing research on the classification and visualization of project tasks, developing an algorithm that can automatically classify text descriptions of sprint tasks, collecting and preparing a training sample of text descriptions of sprint tasks for training and testing the classification model, applying natural language processing methods to improve classification and ensure the accuracy of the results, validating the model on real data to assess the efficiency and accuracy of classification, and analyzing the results. The following **methods** have been used: machine learning methods for classification, text vectorization methods, methods for classifying text descriptions, natural language processing methods, methods for semantic analysis of task description text, methods for processing expert opinions. The following **results** were obtained: a comprehensive approach to using machine learning algorithms, including the collection and processing of textual descriptions of tasks, for classification and involvement of expert opinions to improve the quality of task perception by the project team. Text expressions were classified based on the Bayesian classifier and neural classifiers. A visual representation of the data was implemented. Semantic analysis of the text of the description and title of the tasks was performed. Data markup was obtained to classify the quality of the wording, which was performed by a team of experts. To measure the reliability of the obtained expert assessments, we calculated Cohen's kappa coefficient for each pair of markers. According to the experimental results, the accuracy of the Bayesian classifier is 70%. For the classifier based on deep learning, a neural network for binary classification based on the transformer architecture was selected. The neural network was trained using the Python programming language and deep learning frameworks. The result is a classifier that gives an accuracy score of 83% on a test dataset, which is a good result for a small dataset and data with conflicting labels. **Conclusions:** the analysis of textual data confirms that the existing data in the tracking system is incomplete and contains abbreviations, conventions, and slang. The results show that the assessment of the quality of the wording is determined by the level of expert knowledge of the specifics and context of the project, while increasing the number of experts has almost no effect on the result. In further research, it is recommended to test the hypothesis that the effectiveness of the classifier depends on the specific project and the use of unsupervised learning methods for the task of identifying the quality of formulations.

Keywords: project; task description; project task management system; model; classifier; vector representation.

Introduction

Nowadays, flexible or adaptive management methodologies are widely used to implement IT projects. The use of these methodologies makes it possible to respond quickly to changes and risks that arise during the development of a software product. The effectiveness of the project team is influenced by a large number of factors [1], among which an inaccurate description of the project tasks or their misunderstanding by developers is significant. Thus, identifying ambiguities in sprint task descriptions often becomes critical to successful software development. This allows you to reformulate unclear tasks in a timely manner and avoid misunderstandings between developers.

Tracking systems (also known as task management systems or project management systems) play a key role in IT project management by ensuring that work tasks, resources, deadlines, and communications are effectively organized and tracked. Tracking systems allow you to create and assign tasks to team members. Each task can contain a description, deadlines, attached files, and other important information. Data collection and analysis allows you to create reports and analytical tools to evaluate team performance, identify trends, and make strategic decisions. Thus, project task management systems store information about program tasks and comments that can be useful for predicting the success of tasks. However, at the moment, there are practically no studies on this topic and no results that would allow us to analyze how accurately and clearly the tasks are set.

Assessing the quality and unambiguity of task descriptions is an important part of software development project management. High-quality and accurate task statements allow the project manager and team to better understand the scope of work, resources required, and assess the risks for each task or requirement. It also helps

© M. Grinchenko, M. Rohovyi, 2023

in making decisions about priorities, resource allocation, and balanced project planning.

Text classification plays an important role in the field of natural language processing (NLP) and is becoming a key component in the development and improvement of project tasks. The rapid development of this field allows us to improve the results in project activities. This, in turn, helps to avoid possible misunderstandings, errors, and delays in project implementation. The text classifier allows you to assess the quality of task formulation, determining their clarity and conciseness. The use of classification helps to create a single standard for interpreting textual descriptions of tasks, which facilitates common understanding across the team. In general, the introduction of text classification into the project management process can improve the quality of project tasks, ensuring their more efficient implementation and avoiding possible difficulties in the development process.

Analysis of last achievements and publications

Fixing tasks and describing them in the training system is an important element for managing and controlling project development processes. This practice helps to avoid misunderstandings, set priorities, and determine the progress of each task during the sprint. Brief task notes in natural language allow team members to quickly grasp the essence of tasks. They can include keywords and phrases that identify the main aspects of the tasks.

The project team spends considerable time manually categorizing incoming tasks. Unfortunately, this process is difficult, and team members do not have clear guidelines for best practices based on historical data. As the amount of data grows, it becomes critical to implement automated task classification. Typically, sprint task descriptions are unstructured, making them difficult to process using natural language. Typos and abbreviations in the text of tasks are also complicating factors. Classification of sprint tasks is a critical process that ensures their proper addressing to the appropriate performers.

This paper analyzes studies on the classification and visualization of project tasks. The research can be divided into two groups.

The first group studies the quality problems of describing functional and non-functional requirements. Paper [2] proposes an approach that can be used to identify patterns that meet well-known standards with a measure of F1 of 0.90. Also, this approach allows detecting common syntactic features for non-standard patterns in more than 73.5% of cases. The evaluation showed that these results are reliable regardless of the volume and duration of the processed requirements.

The study of the second group [3] focuses on the differences between vectorization methods that transform requirements written in natural language into numerical vectors for further classification. Through an empirical experiment using 5 open datasets, it was found that advanced vectorization methods significantly improve classification performance. It was also found that using pre-prepared data for vectorization is sometimes the most effective method.

First, let's take a closer look at the studies that belong to the first group.

In [4], the researchers defined the concept of "Requirement Smells" as an indicator of the quality of the requirements description for quick analysis during their formation. The researchers also presented an approach to identifying requirements, which was empirically evaluated in the study of many cases. The paper contains conclusions that relate them to the available evidence on the detection of natural language quality defects in requirements artifacts, and analyzes the impact and limitations of this approach and its evaluation.

Researchers in [5] tried to redefine approaches to measuring standard IEEE metrics (defines the software maintenance process) to improvise a quality assessment of SRS (software requirements specification). The authors recommended an SRS template and a strategy for assessing SRS quality metrics. In contrast to modern methods, this approach brought out the metrics of the IEEE SRS quality assessment standards as more modular for collecting requirements.

In [6], the authors proposed a semi-supervised text categorization approach for the automatic identification and classification of non-functional requirements (NFRs). The goal of this approach is to integrate into a recommender system to help analysts and software developers in the architectural design process. A small number of requirements identified during the discovery process allows learning an initial classifier for NFRs that can consistently identify further requirements for the problem statement in an iterative process.

Paper [7] presents an approach to reducing privacy risks in agile software development by automatically detecting information related to the privacy of user history, which is an important notation in agile requirements engineering (RE). Experimental results

Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 4 (26)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

have shown that deep learning algorithms allow obtaining better predictions than those achieved using conventional (shallow) machine learning methods. The use of Transfer Learning can significantly improve the accuracy of forecasts by up to 10%.

The authors of [16] presented an ontology-based approach to supporting requirement tracking. The developed ontology contains theoretical knowledge for developing requirements in an agile environment, which allows a group of developers to effectively manage the development of requirements for a software product. This facilitates rapid decision-making on revising project artifacts related to changes and allows for a more accurate determination of the scope of these changes.

In the second group of works on the classification and visualization of project tasks, researchers focus on the quality problems of describing user stories in a project.

In [10], two experiments were conducted to investigate the factors that could potentially affect the derivation of static conceptual models from specifications written in the form of user stories and use cases. The analysis of the results showed that the notation of requirements has a limited impact on the quality of the resulting conceptual model in terms of both reliability and completeness. Systematic grammatical analysis leads to conceptual models with a high level of completeness and validity in both experiments.

Using a linguistic classification of ambiguity problems, the authors of [11] offer a new perspective on current knowledge about understanding, avoiding, and resolving ambiguity in user stories. The possibilities of increasing the effectiveness of the user story technique in supporting requirements engineering (RE) activities in application system development (ASD) projects are investigated.

The approach proposed by the researchers [13] consists of a set of rules that receive scattered information in user stories and organize it in Lexicon symbols. The Lexicon provides a consolidated and organized structure to alleviate the problem of confusing information that causes a lack of tracking between different sprints. This approach has advantages for tracking requirements in agile methodologies, which is supported by previous evaluation results.

Study [15] defines a classification of developers' cognitive representation styles and cognitive interaction patterns in agile RE. The cognitive abilities of developers based on their statements when separating and defining user stories were investigated. The conclusions showed that developers tend to focus on the technology of cognitive style of representation, even in RE, and have cognitive difficulties in activating and detailing user stories.

In [17], the authors investigate the effectiveness of using word embedding to classify tasks in the IT help desk. Experimental results indicate that the traditional Total Frequency Inverse Document Frequency (TFIDF) method, combined with the Support Vector Machines (SVM) method, provides competitive results and is sometimes more effective than static methods for building vector representations of words, such as word2vec. The use of TFIDF-SVM allows achieving good results at a low computational cost.

The authors of [9] propose an instrumental approach that combines information visualization with two natural language processing techniques: conceptual model extraction and semantic similarity. Experiments have shown that detecting terminological ambiguities requires considerable time, even with the use of technical means, and that it is difficult to determine whether a synonym can interfere with the correct direction of software system development.

Paper [16] presents a visual analysis method called DeepNLPVis, which is used to understand and analyze the model's behavior in text classification tasks and study the reasons for successful and unsuccessful cases. Studies have shown that task descriptions and comments can be used to predict the success of a task with a high level of accuracy.

In [18], the authors examine the usefulness of textual task descriptions and comments in predicting whether problems will be solved successfully. Experiments have shown that problem descriptions and comments can be used to predict problem solving success with more than 85% accuracy, and that predictions of problem solving success change over time.

In [19], the authors propose a method for classifying tasks by clustering and visualizing the characteristics of each category. This method allows experts and stakeholders to better understand the project and features of the target software system by reviewing the keywords of the task categories.

The analysis of the conducted research suggests that there are approaches to describing tasks and user stories, but the issue of qualitative description of project sprint tasks remains less studied. The need for a high-quality textual description of sprint tasks is especially relevant, since an incorrectly formulated or unclear description can lead to misunderstandings and errors in the execution of tasks.

Unlike assessing the quality of requirements and user stories, assessing the quality of sprint tasks requires taking into account the specifics of this context. At the stage of creating the sprint backlog, it is important to determine how clearly and unambiguously each task is formulated. This will help avoid misunderstandings and incorrect task performance by project team members.

The process of identifying and classifying sprint tasks into "clear" and "unclear" is critical at the stage of backlog formation. This will allow you to identify tasks that may be a source of misunderstanding and risk of project delays due to poor textual description. Classifying tasks will provide the project team with the opportunity to devote special attention and resources to clarifying and improving the description of these tasks, thereby reducing the likelihood of errors and misunderstandings.

Considering the quality and unambiguity of the textual description of sprint tasks as a factor influencing task performance is a key element of effective project management. This helps ensure that tasks are properly understood and effectively completed by team members.

Aim and tasks of the study

In the tracking system, tasks are formulated in such a way as to convey the essence of the work to be done as clearly and understandably as possible. It is important to ensure that each team member understands the task in the same way, to avoid ambiguities and misunderstandings. Experience shows that project executors save information about the task in its title, create descriptions, and add comments. Recording all this information in the tracking system provides the team with a convenient tool for collaboration, simplifies communication, and helps ensure that all project participants understand the tasks accurately and completely.

Thus, the purpose of the study is to find approaches to reduce the risks of failure to complete the tasks of an IT project sprint by automating the classification of text descriptions. To achieve this goal, the following tasks need to be solved:

1. Developing an algorithm that can automatically classify text descriptions of sprint tasks into "clear" and "unclear" according to quality and unambiguity criteria.

2. Collect and prepare a training set of textual descriptions of sprint tasks for training and testing the classification model.

3. Application of advanced natural language processing techniques to improve classification and ensure the accuracy of the results.

4. Validate the model on real data to assess the efficiency and accuracy of the classification.

5. Analyzing the results, identifying the factors that affect the quality of task formulation, and drawing conclusions on the application of the developed model.

Materials and methods

To achieve this goal and develop a model for identifying tasks with poor formulation in an IT project sprint, we propose an integrated approach that includes collecting and processing textual descriptions of tasks, using machine learning algorithms for classification, and involving expert opinions to improve the quality of task perception by the project team.

This study uses a task tracking system as the source of data on sprint tasks. The dataset includes 1000 tasks that were exported from the Jira project management system. Based on data from a real project, this dataset contains information about various task parameters such as name, description, unique key, performer, time spent, time estimate, task status, comments, and others. The language of the received textual descriptions of tasks for wording analysis is English, which is typical for software development projects. The main characteristics for identifying the task wording are the name and description. It should be noted that the description is not a mandatory characteristic of the task, i.e., the task formulation can consist of only the name.

In the process of data pre-processing, a number of cleaning procedures must be performed. In addition to standard textual data parsing, links, code fragments, and uninformative symbols such as currency signs and likes are removed from the dataset. We also replace abbreviations with their full meanings.

The analysis of the dataset reveals the following characteristics: the average number of words in the title is 6.4, and the average number of words in the description is 37.67. The total number of tasks in the set is 1000, of which the number of tasks with descriptions is 461.

The number of unique words in the title and description is 5080. Conjunctions and articles turned out to be the most popular words, and among those that carry an independent semantic load, the most popular are: "add", "image", "inspection", "defect", "object". These words are relevant to this project.

Table 1. *Statistics of the dataset at different stages of cleaning*

Stage	Delete hyperlinks	Delete special characters	Delete word stops	Extension of abbreviations
Number of unique words (title)	1470	1270	1260	1251
Number of unique words (description)	3293	2610	2601	2596
Number of unique words (description + title)	3943	3069	3061	3051
Maximum number of words (title)	18	18	15	16
Average number of words (title)	6.47	6.42	5.62	5.68

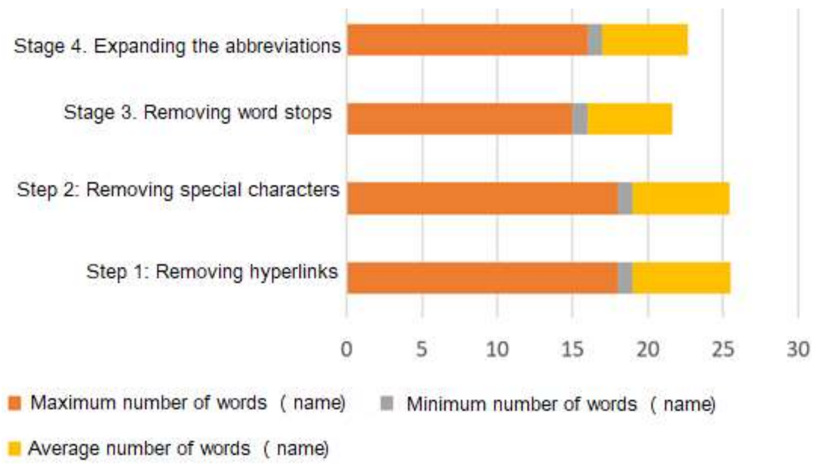

Fig. 1. Statistics of the dataset at different stages of cleaning by the total number of words

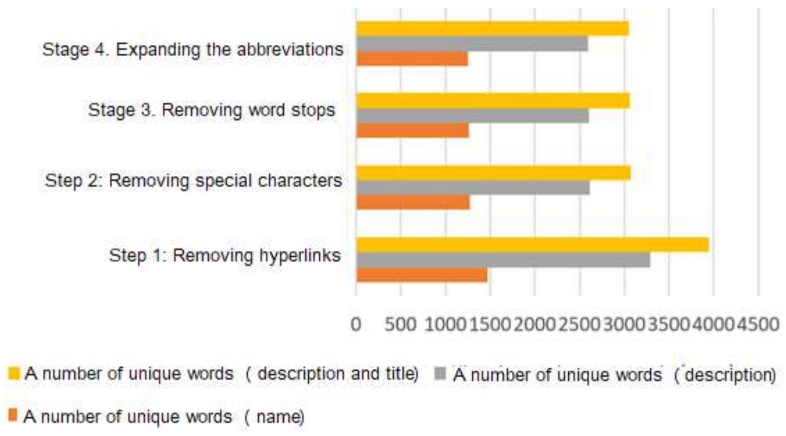

Fig. 2. Statistics of the dataset at different stages of cleaning by the number of unique words

Next, at the second stage of the study, the textual descriptions of the sprint tasks are converted into vector representations for further use in the classification model, which involves obtaining a vector representation of each task. This process starts with converting each task into a set of tokens using a tokenizer. A tokenizer, also known as a lexical analyzer, divides text into small units called tokens [20]. Tokens can be words, symbols, phrases, or even sentences, depending on the type of task and the processing language. Tokenization helps to further represent the text in a format suitable for further analysis and processing by computer algorithms.

There are a variety of approaches to obtaining vector representations of text fragments. These vectors can be generated by using unsupervised learning algorithms or retrained neural networks. Choosing a method of text vectorization is an important research task. The following methods can be used to obtain vector representations:

– Word2Vec – a method of vector representation of words [21];

– TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical method that is widely used in the field of information retrieval and text data analysis [22];

– GloVe (Global Vectors for Word Representation) is a type of word representation using statistical properties of words in large text corpora [23];

– FASTTEXT is a type of vector word representation method based on neural networks and has several unique features that make it popular and effective in natural language processing [24].

At the third stage, tasks are marked up using expert annotations, and two classes are formed. However, key research questions arise: what number of experts is sufficient to ensure an independent and reliable assessment of the quality of task formulation? Can experts' knowledge of the project context influence the results of their expert evaluation? An important aspect is also to determine methods for assessing the consistency of expert opinions and establishing criteria for matching certain classes for further use in training the classification model.

The fourth stage involves training the classifier, where Bayesian classifier and neural classifiers can be effective for classifying text fragments. Determining the optimal classification model that meets the characteristics of the data of a particular project is the task of an experimental study. It is important to note that the classification results will be affected not only by the chosen classification method but also by the approach to text vectorization.

The final stage involves validating the model on the available data and selecting the optimal classifier for further use in other projects. One of the key criteria for evaluating the model is the F-measure. The F-measure is a metric that combines precision and recall into a single numerical score, allowing you to assess the balance between these two considerations. The F-measure provides a trade-off between precision and completeness, which is especially important when dealing with unsorted classes or unbalanced datasets. The overall goal is to achieve the best possible balance between these two aspects so that the model has high accuracy and, at the same time, avoids omissions (missed detections) of a particular class.

This approach to building a classifier allows us to systematize sprint tasks into categories according to the quality of their formulations. In addition, this approach takes into account the project context and expert opinions, which makes it flexible and adaptive to the specifics of a particular project and team characteristics, which is critical for further improving project management processes.

Study results and their discussion

The study was conducted on real data generated from the project management system to automate the identification of poor-quality sprint task formulations that affect the developers' understanding of them and, as a result, lead to project risks. The features of the data collected for the study are semi-structured, incomplete, textual, and lack of project context. Therefore, the data set requires special processing and cleaning.

As mentioned above, for classification tasks, we propose to use textual descriptions of tasks contained in the tracking system. Thus, each task is represented by a title and a more detailed description. In our experiments, we used either only the title or the title concatenated with the description.

In the process of processing the text of the tasks, all titles go through several stages:

– replacing all known abbreviations with their meanings;

– filtering all irrelevant characters (if any);

– normalizing all elements of the task text.

Each task is converted into a set of tokens using a tokenizer that is used by the model and understood by it. For all experiments, we use the Python programming language and deep learning frameworks, including Transformers. Next, we need to obtain a vector representation of the task names. At the initial stage, we used the TF-IDF method, the corpus of which was all the tasks available in the dataset of a real project.

The dataset of 1000 tasks was first analyzed into two classes (positive and negative) according to the expert's opinion on the quality of the wording. Then, using dimensionality reduction tools (PCA and t-SNE), visual representations of the data were obtained (Fig. 3, Fig. 4).

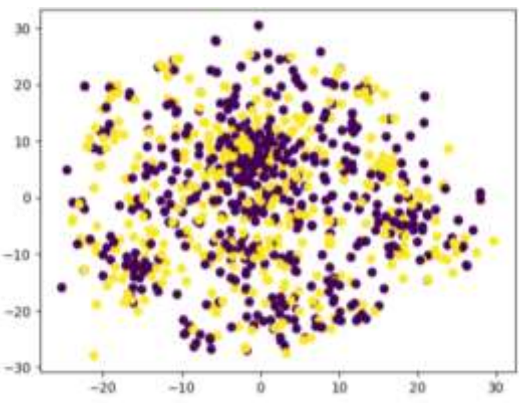

Fig. 3. Visual representation of data using t-SNE in two-dimensional space (yellow – positively annotated tasks, purple – negatively annotated tasks)

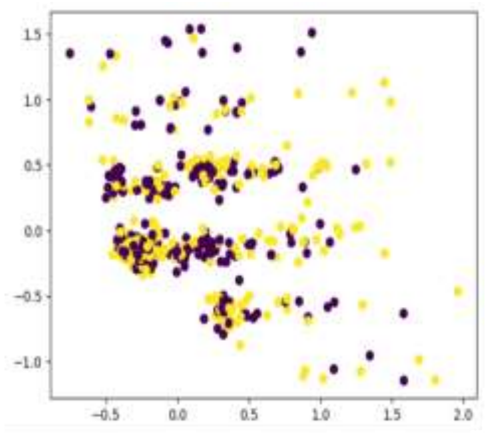

Fig. 4. Visual representation of data using PCA in two-dimensional space (yellow – positively annotated tasks, purple – negatively annotated tasks)

The semantic analysis of the text of the task description and title revealed that the known methods cannot be used due to the lack of context in the description. This is due to the fact that when forming the name and description of a task, the team has its own internal context, which is not transferred to the text of the task. This, in turn, is one of the main reasons for comprehension problems, which affects the timing and quality of the project.

Thus, it has been established that the project context is outside the textual descriptions of tasks, which affects their understanding by new team members, and also limits the use of natural language processing methods to analyze the quality of wording and identify factors that affect the understanding of the content of tasks.

At the next stage of the experiments, additional experts were involved in annotating the dataset. A team of four experts independently labeled the dataset for binary classification. Positive tasks include tasks with high quality, which are understandable to the marker and the time estimate for them is most likely to be adequate. Tasks that are incomprehensible to the expert receive negative marks. In Fig. 5 shows the distribution of positive and negative marks for each of the markers (experts).

It should be noted that when marking up data, as the expert's time and the number of tasks processed increase, the markers become more confident about the project context and a higher percentage of tasks receive positive marks. It is concluded that the more an expert works within the project, the more accurate the estimates become, that is, the clarity of the tasks whose descriptions are recorded in the tracking system significantly depends not only on the wording of their textual description, but also on understanding the content of the project itself. It is assumed that only experts from the middle of the project can provide adequate markup for the data to classify the quality of task formulations.

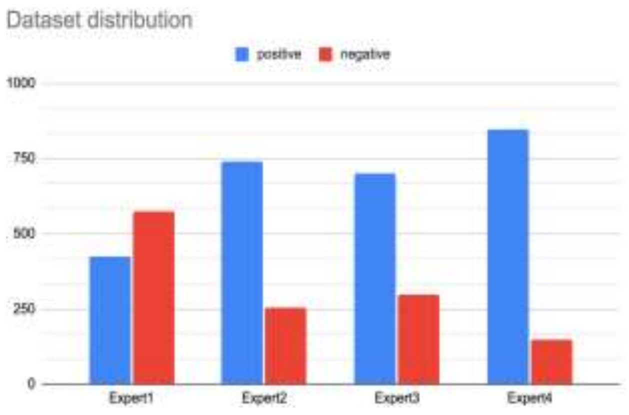

Fig. 5. Distribution of expert markup marks

Thus, we obtained four labeled data sets and analyzed the correlation of the estimates provided by different experts. As can be seen from Fig. 6, the consistency of experts' opinions is low: at best, the markers overlap in only 76% of the tasks, and sometimes only in 43%.

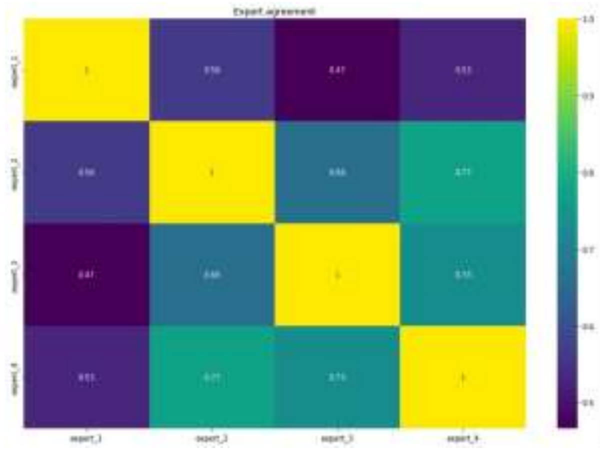

Fig. 6. The level of consistency of the four experts' opinions

Cohen's kappa coefficient is a statistic used to measure inter-rater reliability (as well as intra-rater reliability) for qualitative (categorical) items [25]. It is believed to be a more reliable measure than simply calculating the percentage of agreement, as kappa also takes into account the possibility of random agreement.

Cohen's kappa measures the agreement between two raters, each of whom classifies *N* items into C mutually exclusive categories. The definition of κ is as follows:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} - 1 - \frac{1 - p_0}{1 - p_e} \qquad (1)$$

where p_0 – is the relative observed agreement among raters, and p_e – the hypothetical probability of random agreement, using the observed data to calculate the probability that each observer will randomly see each category.

If the raters are in full agreement, $\kappa = 1$.

If there is no agreement among the raters, other than what would be expected by chance (as indicated by p_e), $\kappa = 0$. It is possible that the statistics will be negative, which may occur by chance if there is no relationship between the estimates of two appraisers, or it may reflect a real tendency of appraisers to give different estimates.

For k categories, N observations to be classified, and n_{ki} is the number of times that evaluator i predicted category k:

$$p_e = \frac{1}{N^2} \sum n_{k1} n_{k2} \qquad (2)$$

In machine learning and statistics, to evaluate binary classifications, the Kappa Cohen formula can be written as follows:

$$\kappa = \frac{2(TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)} \qquad (3)$$

Where TP – true positives;

FP – false positives;

TN – true negatives;

FN – false negatives.

Based on the expert ratings, we calculated Cohen's kappa coefficient for each pair of markers. The results of calculating Cohen's kappa coefficient for the experiment are shown in Fig. 7.

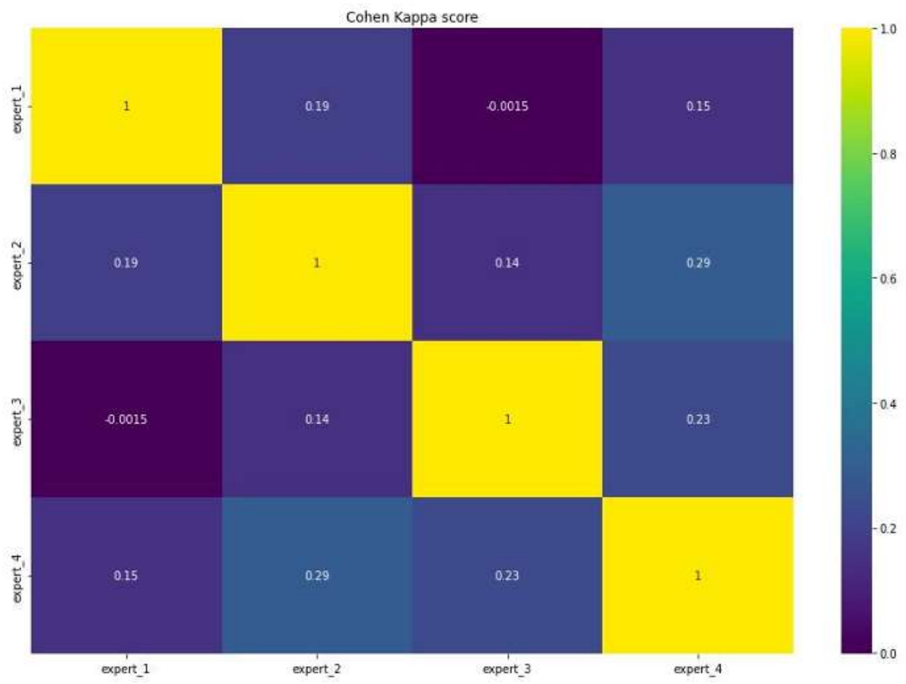

Fig. 7. The Cohen Kappa

Thus, the assumption that task understanding depends on the availability of information about the project, involvement in the development team, etc. was confirmed.

Next, to form the dataset for classification, we selected only those tasks where at least 3 experts gave the same answer about the quality of the task. Thus, we obtained 623 positive and 187 negative task evaluations.

The analysis of modern approaches to automatic text data processing has shown that Bayesian classifier and neural classifiers are well suited for classifying text expressions. A series of experiments were conducted.

Since one of the experiments is training a neural network based on a vectorized representation of the text and, according to the literature, the most representative are the vectors created by artificial neural networks, and given the small amount of data and resources, the trained BERT (bert-base-uncased) model from the Python package for transformer class models was chosen for the experiment. This model was trained on a large dataset, which was tokenized using an appropriate tokenizer, which includes 28996 natural language tokens, 5 of which are the following tokens:

– [PAD] – to add to the sequence to get the required size;

– [UNK] – to replace unknown characters (in this case, this token was checked and was not present in the dataset);

– [CLS] – token to indicate the token in which the class should be located in the original sequence;

– [SEP] – a separator token;

– [MASK] – a masking token used in training with Masked Language Modelling. This training methodology is often used in pre-training natural language models to obtain representative vectors of input text sequences.

Transformers were first introduced in [26] as an encoder-decoder architecture, i.e., the input data is first collapsed to obtain a representation and then unfolded to perform a specific task. The main idea is to use a self-attention mechanism, which allows the model to effectively interact with different parts of the input text simultaneously. The model consists of several layers, each of which contains attention mechanisms and notation (normalization, activation).

The formula for the attention mechanism in Transformers is as follows:

$$Attention(Q,K,V) = softmax\left(QK^T / \sqrt{d_k}\right)V \quad , \tag{4}$$

where Q, K, V – query, key and value matrices, respectively;

d_k – the dimension of the key vector.

Also, it was found that using only one level of attention is less efficient than using the attention mechanism on different linear projections of the query, key, and learned values, then collecting them all together and adding another linear projection to the original representation size. This operation can be described as follows:

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^0, \tag{5}$$

where $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$.

One of the key innovations of the BERT deep learning model, which uses an encoder model that does not perform autoregressive prediction, is two-way contextual encoding, which allows the model to use information from all directions in the text. During training, two types of input representations are used: segmental and positional embeddings.

The formulas for positional and segmental representations in the BERT neural network look like this:

$$Positional = Embedding(w_i) + PositionalEncoding(i), \tag{6}$$

$$Segments = Embedding(s_j), \tag{7}$$

where w_i – token on the position i;

s_j – segment identifier of the sentence j;

$PositionalEncoding(i)$ – is the vector of the position designation i in the input sentence.

There are many options for modeling the position, such as uniformly increasing functions or periodic functions.

The Bayes classifier was chosen as the first classification method. It is based on Bayes' theorem, which determines the probability of an event based on the probabilities of previous events. In the case of text classification, the Bayesian classifier uses the probabilities that a certain word or phrase belongs to a certain class.

The basic idea of a Bayesian classifier is to determine the probabilities for each class, given the occurrence of specific words or phrases in the text. Mathematically, this is expressed as follows:

$$P(C_x|X) = \frac{P(X|C_x) \times P(C_k)}{P(X)} \quad , \tag{8}$$

where $P(C_x|X)$ – is the probability that document X belongs to the class C_k;

$P(X|C_x)$ – is the probability of document X given the class C_k;

$P(C_k)$ – a priori probability of the class C_k;

$P(X)$ – total probability of the document X.

To simplify the calculations, additional assumptions are often used, for example, the independence of word occurrences, which allows expressing the probability of a document given a class as the product of the probabilities of individual words:

$$P(X|C_k) = P(w_1|C_k) \times P(w_2|C_k) \times \ldots \times P(w_n|C_k), \tag{9}$$

where w_i – a single word from a document.

For the Bayesian classifier, vector representations are obtained from the trained BERT neural network. These representations are then used as independent datasets, unrelated to the text, on which the classifier is trained. According to the experimental results, the Bayesian classifier achieved an accuracy of 70%. For the deep learning-based classifier, we chose a neural network for binary classification based on the BERT transformer architecture from the above package. The same model was chosen as a training model. For evaluations, the F1 criterion is used – a metric for measuring the accuracy of models in classification tasks and is a harmonic mean between precision and recall. The result is a classifier that gives an accuracy score of approximately 83% on the test dataset, which is a good result for a small dataset and data with conflicting labels.

The training schedule is shown in Fig. 8.

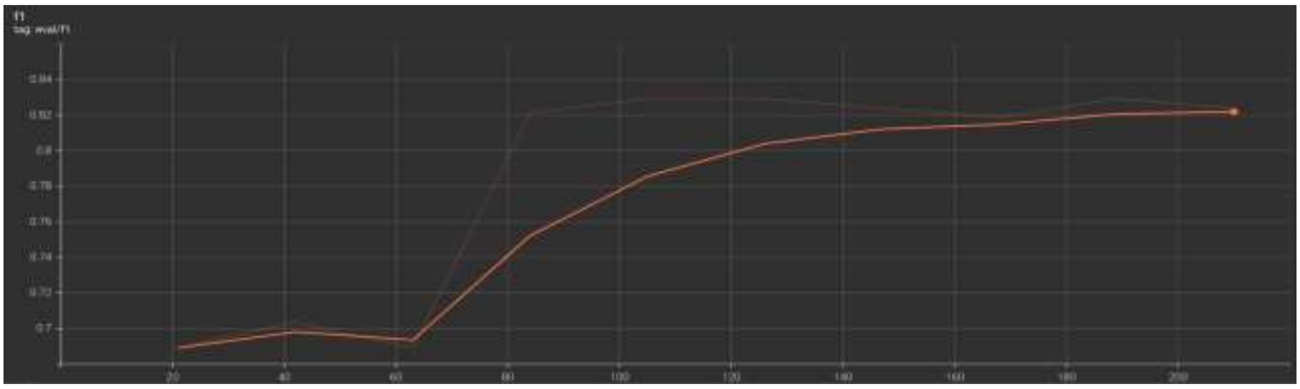

Fig. 8. Graph of F1 metric dependence on the training step

Thus, the experiments have shown that developing a classification model to identify the quality of textual descriptions of sprint tasks requires the involvement of experts who know the project context, determining the vectorization approach, and choosing a classifier model that depends on the available data collected from the project tracking system.

Conclusion and perspectives of further development

Thus, based on the results of the study, the following conclusions can be drawn.

This study focuses on solving the problem of identifying the quality of the textual description of sprint tasks in the project management system, which, in turn, determines an important aspect of trying to reduce the risks of project failure. In solving this problem, the developed and implemented sprint task classifier is used, which automatically identifies poor-quality task formulations. It was found that this can act as a catalyst for improving the formulated tasks and adding detailed descriptions, which contributes to the effective work of the project team as a whole.

The analysis of textual data confirms that the existing data in the tracking system is incomplete and contains abbreviations, symbols, and slang. This undoubtedly makes it difficult to understand. It should be noted that the team understands the project context and is able to perceive the information provided. The results indicate that the quality of the wording is determined by the level of the expert's awareness of the specifics and context of the project, while increasing the number of experts has almost no effect on the result.

Adding formulation assessments during the sprint retrospective to train the classifier model and involve project team members as experts appears to be advisable. This will facilitate data collection for regular model training and improve the consistency of ratings assigned by different experts.

Despite the experiments that did not reveal the superiority of a particular classifier, it is recommended to use several classifiers, compare their results by F-measure, and take into account the choice of vectorizer to select the best one. Experiments on real data from a software development project for a Bayesian classifier and a classifier based on the Transformer architecture showed an accuracy of 0.7 and 0.83, respectively, which is quite acceptable given the training data.

In further research, it is planned to test the hypothesis that the classifier's effectiveness depends on a particular project and the use of unsupervised learning methods for the task of identifying the quality of formulations.

References

1. Rohovyi, M., Grinchenko, M. (2023), "Project team management model under risk conditions". *Vestn. Khar'k. politekhn. in ta. Ser.: Strategichne upravlinnya, upravlinnya portfelyamy, programamy ta proektamy [Bulletin of the Kharkov Polytechnic Institute. Series: Strategic Management, Portfolio Management, Programs and Projects]*, Kharkov: NTU "KhPI", No. 1 (7), P. 3–11. DOI: https://doi.org/10.20998/2413-3000.2023.7.1

2. Sonbol, R., Rebdawi, G., Ghneim, N. (2022), "Learning software requirements syntax: An unsupervised approach to recognize templates, *Knowledge-Based Systems,* Vol. 248, 108933 p. https://doi.org/10.1016/j.knosys.2022.108933

3. Leelaprute, P., Amasaki, S. (2022), "A comparative study on vectorization methods for non-functional requirements classification", *Information and Software Technology,* Vol. 150, 106991 p. https://doi.org/10.1016/j.infsof.2022.106991

4. Femmer, H., Fernández, D., Wagner, S., Eder, S. (2017), "Rapid quality assurance with Requirements Smells", *Journal of Systems and Software*, Vol. 123, P. 190–213. https://doi.org/10.1016/j.jss.2016.02.047

5. Ramesh, M.R.R., Reddy, C.S. (2021), "Metrics for software requirements specification quality quantification", *Computers & Electrical Engineering,* Vol. 96, Part A, 107445 P. 3–11. https://doi.org/10.1016/j.compeleceng.2021.107445

6. Casamayor, A., Godoy, D., Campo, M. (2010), "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach", *Information and Software Technology,* Vol. 52, Issue 4, P. 436–445. https://doi.org/10.1016/j.infsof.2009.10.010

7. Casillo, F., Deufemia, V., Gravino, C. (2022), "Detecting privacy requirements from User Stories with NLP transfer learning models", *Information and Software Technology,* Vol. 146, P. 106853. https://doi.org/10.1016/j.infsof.2022.106853

8. Dalpiaz, F., et al. (2019), "Detecting terminological ambiguity in user stories: Tool and experimentation", *Information and Software Technology*, Vol. 110, P. 3–16. https://doi.org/10.1016/j.infsof.2018.12.007

9. Dalpiaz, F., Gieske, P., Sturm, A. (2021), " On deriving conceptual models from user requirements: An empirical study", *Information and Software Technology,* Vol. 131, 106484 P. 1–13. https://doi.org/10.1016/j.infsof.2020.106484

10. Amna, A.R., Poels, G. (2022), "Ambiguity in user stories: A systematic literature review", *Information and Software Technology,* Vol. 145, P. 1–12. https://doi.org/10.1016/j.infsof.2022.106824

11. Urbieta, M., et al. (2020), "The impact of using a domain language for an agile requirements management", *Information and Software Technology*, Vol. 145, P. 1–16. https://doi.org/10.1016/j.infsof.2020.106375

12. Jia, J., et al. (2019), "Understanding software developers' cognition in agile requirements engineering", *Science of Computer Programming,* Vol. 178, P. 1–19. https://doi.org/10.1016/j.scico.2019.03.005

13. Murtazina, M., Avdeenko, T. (2019), "An Ontology-based Approach to Support for Requirements Traceability in Agile Development", *Procedia Computer Science,* Vol. 150, P. 628–635. https://doi.org/10.1016/j.procs.2019.02.044

14. Y. Wahba, Y., Madhavji, N., Steinbacher, J. (2020), "A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap", *Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering*, P. 151-156. DOI: 10.1109/ICIT58465.2023.10143149

15. Ramírez-Mora, S., Oktaba, H., Gómez-Adorno, H. (2020), "Descriptions of issues and comments for predicting issue success in software projects", *Journal of Systems and Software*, Vol. 168, P. 1–19. https://doi.org/10.1016/j.jss.2020.110663

16. Li, Z., A "Unified Understanding of Deep NLP Models for Text Classification", available at: https://arxiv.org/abs/2206.09355 (last accessed 08.11.2023).

17. Ishizuka, R., et al. (2022), "Categorization and Visualization of Issue Tickets to Support Understanding of Implemented Features in Software Development Projects", *Applied Sciences*. № 12(7):3222. https://doi.org/10.3390/app12073222

18. Devlin, J., et al. (2019), "BERT: Pre-training of deep bidirectional transformers for language understanding", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, P. 4171–4186. DOI:10.18653/v1/N19-1423

19. Chawla, P., Hazarika, S., Shen, H.-W. (2020), "Token-wise sentiment decomposition for convnet: Visualizing a sentiment classifier", *Visual Informatics*, Vol. 4 Issue 2, P. 132–141. https://doi.org/10.1016/j.visinf.2020.04.006

20. Bird, S., Klein, E., Loper, E. "Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Beijing", 2009. 504 p. available at: https://tjzhifei.github.io/resources/NLTK.pdf (last accessed 08.11.2023).

21. "Word2vec", available at: https://www.tensorflow.org/text/tutorials/word2vec (last accessed 08.11.2023).

22. "TF-IDF (Term Frequency-Inverse Document Frequency)", available at: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/i (last accessed 08.11.2023).

23. Pennington, J., Socher, R., Manning, C. (2014), "GloVe: Global Vectors for Word Representation", *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, P.1532–1543. http://dx.doi.org/10.3115/v1/D14-1162

24. "Open-source FastText", available at: https://fasttext.cc/ (last accessed 08.11.2023)

25. McHugh, Mary L. (2012), "Interrater reliability: the kappa statistic", *Biochemia Medica*, Vol. 22 Issue 3, P. 276–282. https://doi.org/10.11613/BM.2012.031

26. Ashish Vaswani, Noam Shazeer, Niki Parmar et al., (2017), "Attention Is All You Need", *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, P.1–15. DOI: https://doi.org/10.48550/arXiv.1706.03762

Received 04.12.2023

Відомості про авторів / About the Authors

Гринченко Марина Анатоліївна – кандидат технічних наук, доцент, Національний технічний університет "Харківський політехнічний інститут", завідувачка кафедри стратегічного управління, Харків, Україна; e-mail: marinagrunchenko@gmail.com; ORCID ID: https://orcid.org/0000-0002-8383-2675

Роговий Микита Антонович – Національний технічний університет "Харківський політехнічний інститут", аспірант кафедри стратегічного управління, Харків, Україна; ORCID ID: https://orcid.org/0000-0002-7902-3592; e-mail: nikrogovoy@gmail.com

Grinchenko Marina – PhD (Engineering Sciences), Associate Professor, National Technical University "Kharkiv Polytechnic Institute", Head at the Department of Strategic Management, Kharkiv, Ukraine.

Rohovyi Mykyta – National Technical University "Kharkiv Polytechnic Institute", PhD student at the Department of Strategic Management, Kharkiv, Ukraine.

МОДЕЛЬ ІДЕНТИФІКАЦІЇ ЗАВДАНЬ СПРИНТУ ПРОЄКТУ НА ОСНОВІ ЇХ ОПИСУ

Предметом дослідження є ідентифікація завдань спринту проєкту. **Мета статті** – пошук підходів до зниження ризиків невиконання завдань спринту. У роботі вирішуються такі **завдання**: аналіз досліджень щодо класифікації та візуалізації завдань проєкту; розроблення алгоритму, який здатний автоматично класифікувати текстові описи завдань спринту; збір і підготовка навчальної вибірки текстових описів завдань спринту для навчання та тестування моделі класифікації; застосування методів оброблення природної мови для вдосконалення класифікації та забезпечення точності результатів, проведення валідації моделі на реальних показниках для оцінювання ефективності й точності класифікації; проведення аналізу результатів. Використовуються такі **методи**: машинне навчання для класифікації, векторизація текстів, класифікація текстових описів, оброблення природної мови, семантичний аналіз тексту опису завдань та оброблення експертних оцінок. **Досягнуті результати**. Запропоновано комплексний підхід використання алгоритмів машинного навчання, що передбачає збір та оброблення текстових описів завдань, для класифікації та залучення експертних оцінок з метою вдосконалення якості сприйняття завдань командою проєкту. Проведено класифікацію текстових висловів на основі класифікатора Баєса та нейронних класифікаторів. Реалізовано візуальну репрезентацію даних. Проведено семантичний аналіз тексту опису та назви завдання. Отримано розмітку даних для класифікації якості формулювань, яка була виконана командою експертів. Для вимірювання надійності отриманих оцінок експертів розраховано коефіцієнт каппа Коена для кожної пари розмітників. За результатами експериментів для класифікатора Баєса отримано точність 70%. Для класифікатора на основі глибокого навчання обрано нейронну мережу для бінарної класифікації на основі архітектури *transformer*. Проведено навчання нейронної мережі за допомогою мови програмування *Python* і фреймворків для глибокого навчання. Унаслідок отримано класифікатор, що на тестовому наборі оцінює з точністю 83%, що є гарним результатом для малого набору даних і даних із суперечливими мітками. **Висновки.** Аналіз текстової інформації підтверджує, що наявні в системі трекінгу дані не повні та містять скорочення, умовні позначки та сленг. Здобуті результати свідчать про те, що оцінка якості формулювань визначається рівнем обізнаності експерта щодо особливостей і контексту проєкту, водночас збільшення кількості експертів майже не впливає на результат. У подальших дослідженнях рекомендується перевірити гіпотезу про залежність ефективності класифікатора від конкретного проєкту та використання методів навчання без учителя для завдання ідентифікації якості формулювань.

Ключові слова: проєкт; опис завдань; система управління завданнями проєктів; модель; класифікатор; векторна репрезентація.

Бібліографічні описи / Bibliographic descriptions

Гринченко М. А., Роговий М. А. Модель ідентифікації завдань спринту проєкту на основі їх опису. *Сучасний стан наукових досліджень та технологій в промисловості*. 2023. № 4 (26). С. 33–44. DOI: https://doi.org/10.30837/ITSSI.2023.26.033

Grinchenko, M., Rohovyi, M. (2023), "A model for identifying project sprint tasks based on their description", *Innovative Technologies and Scientific Solutions for Industries*, No. 4 (26), P. 33–44. DOI: https://doi.org/10.30837/ITSSI.2023.26.033