DOI: 10.17587/prin.13.322-330

Applying Supervised Machine Learning Methods to Determine the Atomicity of Requirements for Complex Technical Systems[1]

K. I. Gaydamaka, k.gaydamaka@gmail.com, Department of Systems Engineering, "MIREA — Russian Technological University", Moscow, 119454, Russian Federation,
P. A. Ognyanovich, Pasha2la71@gmail.com,
National Research Nuclear University "MEPhI"

Corresponding author:

Kirill I. Gaydamaka, Senior Lecturer, Department of Systems Engineering, "MIREA — Russian Technological University", Moscow, 119454, Russian Federation
E-mail: k.gaydamaka@gmail.com

Received on November 16, 2021
Accepted on June 06, 2022

The article is devoted to the problem of determining the atomicity of requirements for complex technical systems. The purpose of this article is to apply supervised machine learning methods, namely classification, to determine the atomicity of requirements. It is assumed that feature engineering based on the linguistic features of requirements statements will make it possible to classify requirements into atomic and non-atomic ones with high accuracy. The article describes the use of the following methods for generating features: fastText, BERT, customFeature. Requirements are classified using a logistic classifier, decision trees, random forest, and gradient boosting. The best combination of methods turned out to be using customFeature with gradient boosting.

Keywords: *gradient boosting, BERT, requirements management, machine learning, requirements atomicity, requirements quality*

For citation:

Gaydamaka K. I., Ognyanovich P. A. Applying Supervised Machine Learning Methods to Determine the Atomicity of Requirements for Complex Technical Systems, *Programmnaya Ingeneria*, 2022, vol. 13, no. 7, pp. 322—330.

УДК 004.852

К. И. Гайдамака, ст. преподаватель, k.gaydamaka@gmail.com,
РТУ МИРЭА,
П. А. Огнянович, студент, Pasha2la71@gmail.com,
Национальный исследовательский ядерный университет "МИФИ"

Применение методов машинного обучения с учителем для определения атомарности требований к сложным техническим системам

Статья посвящена проблеме определения атомарности требований к сложным техническим системам. Цель этой статьи — исследовать возможность применения методов машинного обучения с учителем, в частности, классификации, для определения атомарности требований. Предполагается, что разработка признаков, основанная на лингвистических особенностях формулировок требований, позволит с высокой точностью классифицировать требования

[1] The article is based on the materials of the report at the Seventh International Conference "Actual problems of Systems and Software Engineering" APSSE 2021.

на атомарные и неатомарные. Описано использование следующих методов формирования признаков: fastText, BERT, customFeature. Требования классифицируются с использованием логистического классификатора, метода деревьев решений, случайного леса и градиентного бустинга. Экспериментальное исследование показало, что наилучшей комбинацией методов является сочетание customFeature с градиентным бустингом.

Ключевые слова: *градиентный бустинг, BERT, управление требованиями, машинное обучение, атомарность требований, качество требований*

Introduction

The development of high-quality system requirements plays a decisive, key role in the process of working on a project, regardless of its scale [1]. Time, development cost and other indicators of the project's effectiveness directly depend on how accurately and correctly, the system requirements describe the system that must to be created. Taking into account the fact, how complex, large-scale projects can be, it would be logical to assert that the number of requirements in them can amount to several thousand [2]. And since the development of requirements is entirely on the shoulders of the requirements engineers, it would be logical to assume also that it should not depend on the human factors: inattention, negligence or haste. But in cases where huge sums of money are at stake, and due to the imperfection of the human resources, the project executor may suffer significant losses, it is necessary to minimize the human factor.

The requirements make it possible to determine what the interested parties want to get from the system under development and what properties the system should have in order to satisfy their needs.

The quality of the requirements can be judged by the presence of several quality criteria. There are certain criteria that every statement of requirement should meet. These are summarised as follows [3]:

- *Atomic*: each statement carries a single traceable element;
- *Unique*: each statement can be uniquely identified;
- *Feasible*: technically possible within cost and schedule;
- *Legal*: legally possible;
- *Clear*: each statement is clearly understandable;
- *Precise*: each statement is precise and concise;
- *Verifiable*: each statement is verifiable, and it is known how;
- *Abstract*: does not impose a solution of design specific to the layer below.

In addition, there are other criteria that apply to the set of requirements as a whole:

- *Complete*: all requirements are present;
- *Consistent*: no two requirements are in conflict;
- *Non-redundant*: each requirement is expressed once;
- *Modular*: requirements statements that belong together are close to one another;
- *Structured*: there is a clear structure to the requirements document;
- *Traceable*: the appropriate degree of traceability coverage has been achieved.

The article is devoted to the problem of determining the atomicity of requirements for complex technical systems. The purpose of this article is to apply supervised machine learning methods, namely classification, to determine the atomicity of requirements.

Description of the dataset

The initial data is a table of 287 records with the following columns: "ID", "Requirement" and "Singular", where "ID" is the unique identification number of the requirement, "Requirement" is the textual wording of the requirement statement, and "Singular" is the tag of membership of each requirement in the class of atomic or non-atomic (1 and 0, respectively). All requirements statements in dataset are in Russian.

Feature engineerning

There is one very interesting principle in computer science: garbage in, garbage out (GIGO). That is, if incorrect data is fed to the algorithm, the results will be incorrect, even if the algorithm is the correct and accurate. Supervised machine learning algorithms are no exception [4]. Therefore, data preprocessing and feature engineering are very important steps in machine learning. Consider two neural network-based natural language processing models: fasttext and BERT, and custom created features.

FastText is a library developed by Facebook that contains pre-trained ready-made vector representations of words [5]. The algorithm uses both CBOW (continuous bag of words) and skipgram models. For us, the main interest is that FastText does not require training. There are ready-made models for 157 languages, including Russian. By default FastText uses 300 word vectors, but this can be changed if desired. However, FastText

converts a word into a vector, and we have the text of the requirement as the initial data. The main idea of using FastText to transform a textual formulation of a requirement into a vector representation is that the requirement is cleared of all special characters and signs, tokenized and lemmatized [6], after which a list of tokens is fed to the FastText model input. Further, the values of the obtained set of vector representations of individual words of the requirement formulation are averaged.

The BERT model was announced and made public in 2018 [7]. Initially, the BERT model is trained on a huge amount of text taken from books, Wikipedia, etc., and in several languages. This technology is based on such algorithms developed by the NLP (natural language processing) community as learning with partial involvement of a teacher, ELMo models (Embeddings from Language Models) [8], ULMFiT (Universal Language Model Fine-Tuning) [9], OpenAI Transformer and others.

BERT is based on the Transformer, in fact, BERT is a trained stack of Transformer encoders. Encoders are objects of the same structure, but with different weights. Each encoder consists of two parts — an inner understanding layer and a propagation layer. The encoder receives as input numerical vectors obtained from words, processes the vectors at the level of the internal understanding layer and transmits them to the feedforward neural network. The result of the output of one encoder level is the input for the next encoder. Encoders form a stack of 12 (24 for deeply trained models) elements, an encoder stack, and is the backbone of BERT, as mentioned. Each sent word in BERT goes through a chain of encoders and as a result, for each word, a numeric vector of dimension 768 is formed for the base BERT implementation. The obtained vectors are the required ones in the framework of this problem. Further, it is possible to perform various operations with them: cluster, use as input data in supervised machine learning algorithms, for example, for a logistic regression problem.

The customFeature method is based on the use of the language features of the requirements statement. How can an atomic requirement differ from a non-atomic requirement? The first thing that comes to mind is the size of the requirement [10]. It is logical that it is unlikely that a large requirement in most cases will be atomic. On the other hand, you can pay attention to the number of words in the requirement, since the number of characters alone cannot provide enough information. You can also count the number of offers in a request. In most cases, a requirement that consists of more than one clause is not atomic.

Next, it's worth remembering what an atomic requirement is and taking a closer look at the differences between an atomic and a non-atomic requirement. A requirement is considered atomic if each statement of the requirement must be one element of the hierarchy of requirements, and the subsequent division of the proposal into two or more requirements is not possible. Consider the following examples.

- The service life of the phone must be at least 1 year.
- The service life of the phone and its individual parts must be at least 1 year.

Obviously, the second requirement is non-atomic, in contrast to the first, because it can be divided into two requirements: "The service life of the phone must be at least 1 year" and "The service life of individual parts of the phone must be at least 1 year". So what are the fundamental differences between them in terms of language features?

Let's take a look at the INCOSE Requirements Writing Guide [11]. It contains rules for statements and sets of requirements that will be very useful for the feature engineering process. It clearly states: avoid conjunctions in requirements. In the example of a non-atomic requirement, the conjunction "and" is just used, which means it is logical to make a sign that determines the total number of unions in the checked requirement.

In Russian, the enumeration can be specified not only with the help of conjunctions, but also with the use of punctuation marks. The presence of a large number of special characters may indicate a possible non-atomicity of the requirement, so it is worth counting either their total number, or each special character separately.

The next rule that should definitely be used to form a feature is to formulate a requirement in one sentence. Indeed, if it was necessary to split the text into two sentences, then it is worth thinking about splitting the requirement itself into two. Therefore, you should define a binary sign that signals the fulfillment of this rule.

In addition, the INCOSE guidance prescribes not to use brackets in the wording of requirements, not to indicate clarifying text in brackets. Often, information in brackets indicates information that is redundant, deprives the requirement of unambiguity, or contains enumerations. To determine whether the rule described above is followed, it is worth introducing a binary feature that takes the value of one if there are round, square or angle brackets in the requirement.

Also in the INCOSE manual it is written that when writing requirements, you should use the active voice. That is, the requirement "The identity of the client must be verified" is unacceptable. In a simple case, as in the example, the passive voice can be determined by the presence in the text of the requirement of the con-

struction "must be", but due to the peculiarities of the Russian language, one should also parse the sentence and check that the predicate is in the active voice. To check this rule, it is necessary to select a binary sign that signals the presence of a passive voice in a demand.

Another interesting rule is to avoid imprecise terms such as "several", "any", "acceptable", "many", etc. Adverbs are necessary to clarify an action, and not vice versa. In addition, you should not use ambiguous adjectives, the use of which introduces ambiguity: effective, relevant, sufficient, adequate, etc. To implement this rule, you can enter a list of inaccurate words and for each requirement calculate the proportion of such words.

Also, in the formulation of requirements, it is necessary to avoid using the particle "not". Consider an example of a requirement: "The system must not release hazardous substances into the atmosphere." The problem with the formulation of this requirement is that it is impossible to verify its implementation in a finite time. The existence of this rule leads us to the formation of a binary feature that determines the presence of the "not" particle in the checked requirement.

The presence of a slash "/" in the statement of requirements is also undesirable. A typical example — and/or, allows for ambiguity in interpretation. An exception is the forward slash in SI units, for example, km/h, m/s.

Another interesting rule is to avoid pronouns. The use of pronouns in order to avoid repetitions in the formulation of requirements is unacceptable, since it entails the appearance of ambiguity, and therefore problems when verifying the implemented requirement. This rule can be interpreted as a quantitative feature, the value of which will be the proportion of pronouns in the wording of the requirement under consideration.

In addition, absolute phrases that denote an unattainable goal should not be used in the statement of requirements. The indicators of non-fulfillment of this rule can be the presence in the formulation of the requirement of such words as "all", "always", "never", etc. Such requirements cannot be verified. Hence, a quantitative feature appears, which will be equal to the fraction of words of absolute revolutions in the formulation of the requirement under consideration.

Thus, the customFeature feature space is formed, onto which you can map any text formulation of the requirement and obtain a vector from this feature space.

Atomicity check

In the general case, the classification problem can be described as follows: there is a certain sample of objects for which the belonging to the class is determined by labels [12]. The set of classes is finite and known. It is required to implement an algorithm that will be able to determine the class of the entire set of objects without labels.

The problem of determining the atomicity of a requirement formulation can be formulated as a classification problem: the set of classes is represented by two elements: atomic, not atomic. Adapting the formulation of the classification problem to the problem under consideration, we obtain the following statement: it is necessary to build an algorithm that will perform the binary classification of the requirement.

Since the classification problem is a special case of supervised learning, it becomes necessary to compose and mark up a training sample. For this, a requirements specification of 287 elements was formed and marked up. The ratio of elements of different classes in the sample is 43 to 57, which makes it possible to assert that it is balanced. The specification of requirements was divided in a ratio of 3 to 7 — into test and training samples, respectively.

Let's take a closer look at gradient boosting. It is a composition of N basic algorithms, constructed into a composition [13]. They are built sequentially, one after the other, so that the i-th algorithm corrects the composition errors of the previous i-1 pieces. Because of this method of building a composition, it becomes possible to use relatively simple basic algorithms, for example, shallow trees. Sometimes a constant classifier can be taken as the first basic algorithm, which assigns all objects to one class.

Let us consider in order all variants of combinations of classifiers and requirements preprocessing. Let's start with a logistic classifier [14] with various options for preprocessing the initial data in text format.

As an evaluation metrics we will use Acuracy and the area under the ROC curve [15].

Figure 1 shows the results of evaluating the logistic classifier and customFeature.

The figure shows that the logistic classifier trained on the requirements converted into vectors using customFeature showed an accuracy of 86 % on the test sample, and the area under the ROC curve was 0.853. Next, consider the results obtained for the logistic classifier with FastText preprocessing (fig. 2).

The results turned out to be much more modest than those of the previous data preprocessing method. Only 75 % accuracy and 0.741 area under the AUC ROC curve. The next in line will be a logistic classifier using BERT. The results of the assessment are shown in fig. 3.

It can be seen that the accuracy when using BERT is comparable to the first option, 83 %. The area under the ROC curve is 0.832.

Next, consider classifiers based on decision trees. Let's start by combining the decision tree and the customFeature (fig. 4).

	precision	recall	f1-score	support
0	0.84	0.91	0.87	45
1	0.89	0.79	0.84	39
accuracy			0.86	84
macro avg	0.86	0.85	0.86	84
weighted avg	0.86	0.86	0.86	84

ROC AUC: 0.8529914529914528

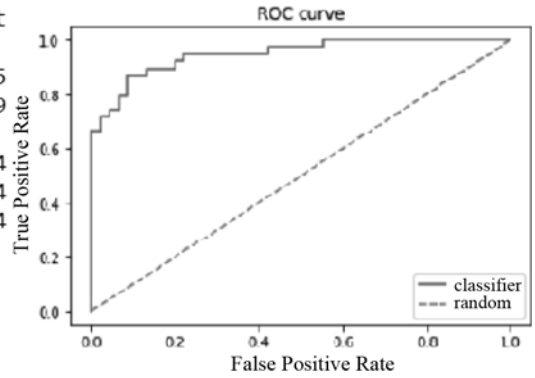

Fig. 1. Logistic + customFeature Results

	precision	recall	f1-score	support
0	0.72	0.87	0.79	45
1	0.80	0.62	0.70	39
accuracy			0.75	84
macro avg	0.76	0.74	0.74	84
weighted avg	0.76	0.75	0.75	84

ROC AUC: 0.7410256410256411

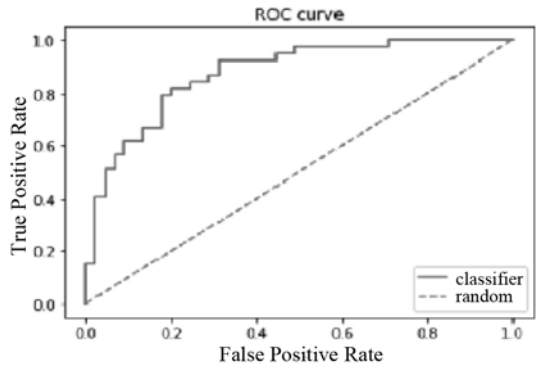

Fig. 2. Results of Logistic + FastText

	precision	recall	f1-score	support
0	0.84	0.84	0.84	45
1	0.82	0.82	0.82	39
accuracy			0.83	84
macro avg	0.83	0.83	0.83	84
weighted avg	0.83	0.83	0.83	84

ROC AUC: 0.8324786324786325

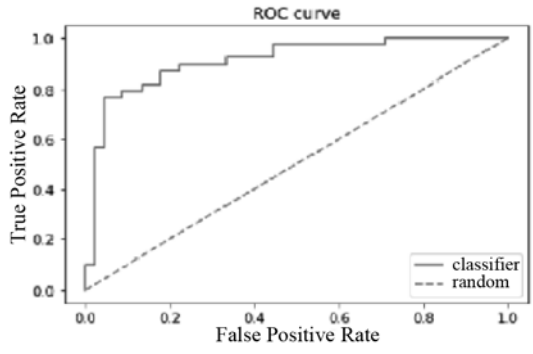

Fig. 3. Results of Logistic + BERT

	precision	recall	f1-score	support
0	0.83	0.87	0.85	45
1	0.84	0.79	0.82	39
accuracy			0.83	84
macro avg	0.83	0.83	0.83	84
weighted avg	0.83	0.83	0.83	84

ROC AUC: 0.8307692307692308

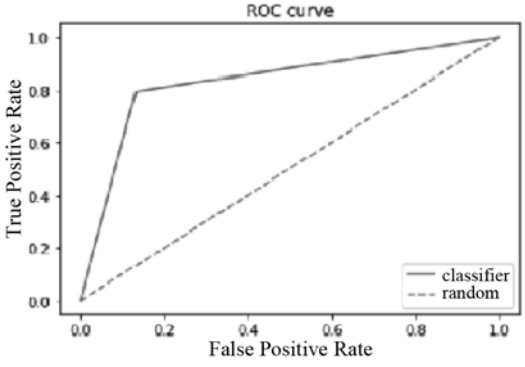

Fig. 4. Decision Tree + customFeature Results

The decision tree using customFeature showed a slightly more modest result than the logistic classifier: the accuracy is 83 % and the area under the ROC curve is 0.831. Let's move on to the next combination - decision tree and FastText. The results are shown in fig. 5.

The result is the worst result among all those tested. Only 69 % accuracy and an area under the ROC curve is 0.684. Next is the decision tree and BERT (fig. 6).

This combination received an accuracy rating of 76 % and an area under the ROC curve of 0.764. At the moment, we would like to recall the peculiarity of decision trees to unnecessarily adjust to the training set, and as a result, have poor generalizing ability, which we saw in the last two combinations due to the large dimension of the vector space (300 for FastText and 768 for BERT).

Let's move on to considering a random forest. Let's start with the customFeature for preprocessing the initial data (fig. 7).

So far we get the highest scores: 88 % accuracy and 0.875 area under the ROC curve. Let's look at more complex preprocessing methods, first FastText. The results are shown in fig. 8.

Again we run into rather low results when using fastText: 80 % accuracy and 0.784 area under the ROC curve. Next up is a random forest combined with BERT. The results are shown in fig. 9.

	precision	recall	f1-score	support
0	0.69	0.78	0.73	45
1	0.70	0.59	0.64	39
accuracy			0.69	84
macro avg	0.69	0.68	0.68	84
weighted avg	0.69	0.69	0.69	84

ROC AUC: 0.6837606837606839

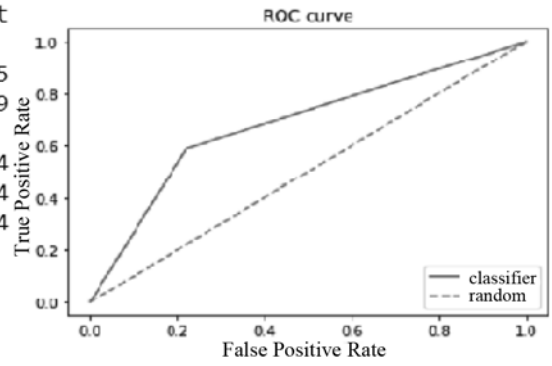

Fig. 5. Decision Tree + FastText Results

	precision	recall	f1-score	support
0	0.80	0.73	0.77	45
1	0.72	0.79	0.76	39
accuracy			0.76	84
macro avg	0.76	0.76	0.76	84
weighted avg	0.77	0.76	0.76	84

ROC AUC: 0.764102564102564

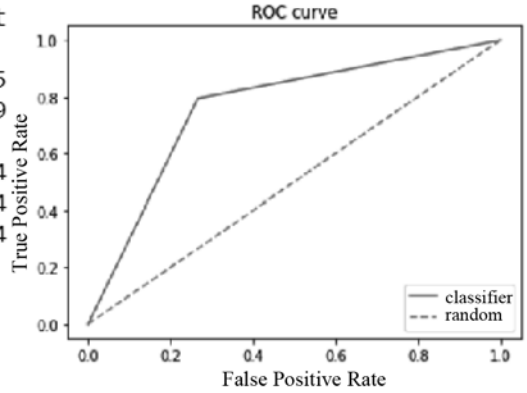

Fig. 6. Results of Decision Tree + BERT

	precision	recall	f1-score	support
0	0.84	0.96	0.90	45
1	0.94	0.79	0.86	39
accuracy			0.88	84
macro avg	0.89	0.88	0.88	84
weighted avg	0.89	0.88	0.88	84

ROC AUC: 0.8752136752136751

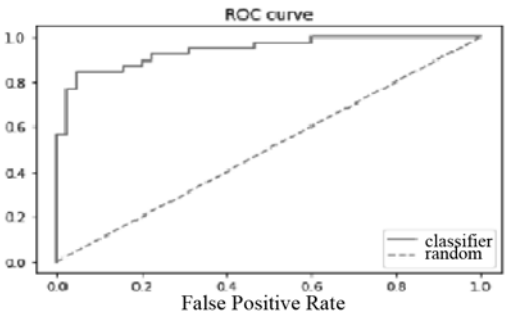

Fig. 7. Random Forest + customFeature Results

	precision	recall	f1-score	support
0	0.73	0.98	0.84	45
1	0.96	0.59	0.73	39
accuracy			0.80	84
macro avg	0.85	0.78	0.78	84
weighted avg	0.84	0.80	0.79	84

ROC AUC: 0.7837606837606839

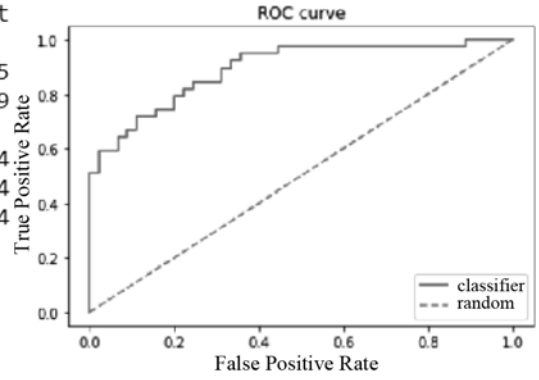

Fig. 8. Results of Random Forest + FastText

	precision	recall	f1-score	support
0	0.83	0.96	0.89	45
1	0.94	0.77	0.85	39
accuracy			0.87	84
macro avg	0.88	0.86	0.87	84
weighted avg	0.88	0.87	0.87	84

ROC AUC: 0.8623931623931624

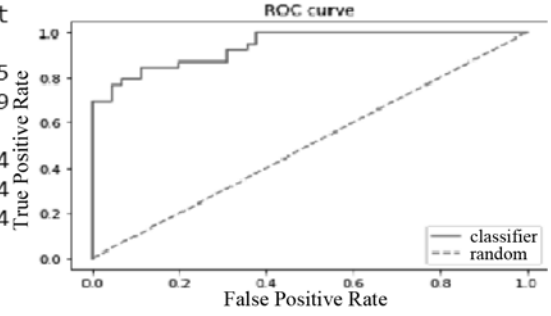

Fig. 9. Results of Random Forest + BERT

Random forest in combination with BERT showed fairly high scores: 87 % accuracy and an area under the ROC curve of 0.862. Continuing the consideration of the results, the next step is a classifier based on gradient boosting over decision trees. First, let's take a look at the customFeature results (fig. 10).

Again we see excellent results for customFeature: 88 % accuracy combined with 0.874 area under the ROC curve. The next step is the results of preprocessing using FastText (fig. 11).

The FastText results are also modest this time against the background of other data preprocessing methods: an accuracy of 81 % and an area under the ROC curve of 0.803. Let's consider the last option — gradient boosting over decision trees in combination with BERT. The results of this option are shown in fig. 12.

The last method of checking the requirements for atomicity showed the following results: an accuracy of 83 % and an area under the ROC curve of 0.829.

	precision	recall	f1-score	support
0	0.83	0.98	0.90	45
1	0.97	0.77	0.86	39
accuracy			0.88	84
macro avg	0.90	0.87	0.88	84
weighted avg	0.89	0.88	0.88	84

ROC AUC: 0.8735042735042734

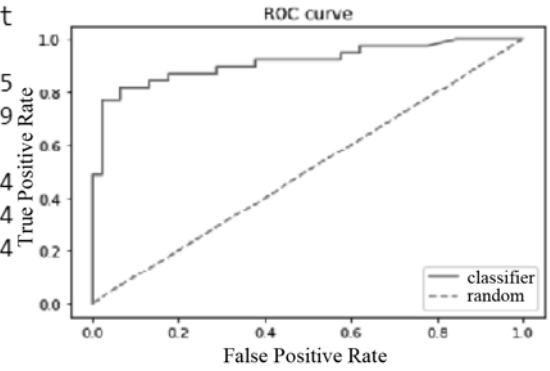

Fig. 10. Gradient Boosting + customFeature Results

	precision	recall	f1-score	support
0	0.78	0.89	0.83	45
1	0.85	0.72	0.78	39
accuracy			0.81	84
macro avg	0.82	0.80	0.81	84
weighted avg	0.81	0.81	0.81	84

ROC AUC: 0.8034188034188035

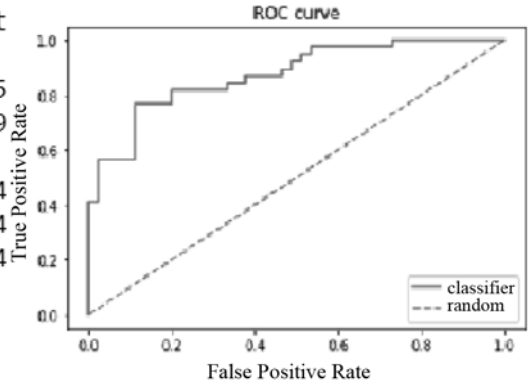

Fig. 11. Gradient Boosting + FastText Results

	precision	recall	f1-score	support
0	0.82	0.89	0.85	45
1	0.86	0.77	0.81	39
accuracy			0.83	84
macro avg	0.84	0.83	0.83	84
weighted avg	0.84	0.83	0.83	84

ROC AUC: 0.8290598290598289

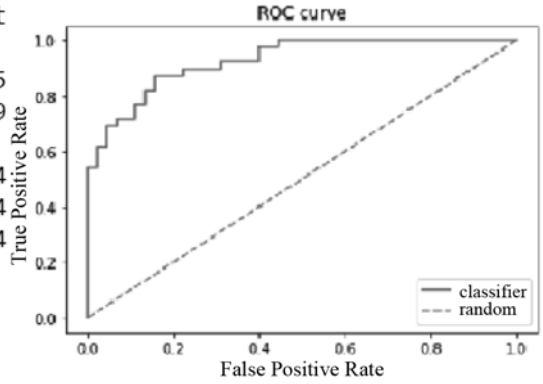

Fig. 12. Gradient Boosting + BERT Results

General results of combinations of requirements verification methods

Learning method	customFeature		Fast Text		BERT	
	Accuracy, %	ROC	Accuracy, %	ROC	Accuracy, %	ROC
Logistic classifier	86	0.853	75	0.741	83	0.832
Decision tree	83	0.831	69	0.684	76	0.764
Random forest	88	0.875	80	0.784	87	0.862
Gradient boosting	88	0.874	81	0.803	83	0.829

Conclusion

To summarize, what is the best combination, what is better to use when checking sets of requirements for atomicity? Consider the results of all combinations of methods in table.

Comparison of the results obtained when testing combinations of data preprocessing methods and algorithms for classifying requirements for atomicity allow us to draw several conclusions:

Using FastText as a way to vectorize a requirement statement has proven to be the least accurate option.

BERT's accuracy is comparable to customFeature using any of the possible classification methods.

Maximum accuracy of 88 % is achieved when using customFeature in combination with random forest or gradient boosting.

The results obtained allow us to make the assumption that models based on a small number of meaningful features, which are important precisely in determining the atomicity of requirements when classifying requirements, turn out to be no worse than those that take into account the semantics of the proposal (customFeature in comparison with BERT).

Thus, the best way to classify requirements for atomicity is gradient boosting over decision trees using the customFeature vector representation of requirements.

The result obtained can be used to create systems for automatic quality control of requirements. The requirements engineer will be able to get quick feedback on the quality of the requirements and not have to wait for the requirements review. The use of machine learning allows to avoid setting up rules for checking the quality of requirements. Instead, it is enough to have a labeled dataset.

References

1. **Chatzipetrou P., Unterkalmsteiner M., Gorschek T.** Requirements' Characteristics: How do they Impact on Project Budget in a Systems Engineering Context? *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* 2019 Aug. 28, IEEE, 2019, pp. 260—267.
2. **Batovrin V., Gaydamaka K.** Requirements engineering — key factor for project success, *The Project Management Journal*, 2017, no. 1 (49), pp. 6—20.
3. **Hall E.** *Requirements Engineering*, US, Kent, Gray Publishing, 2005, 239 p.
4. **Muhamedyev R.** Machine learning methods: An overview, *Computer modelling & new technologies*, 2015, vol. 19, no. 6, pp. 14—29.
5. **Joulin A., Grave E., Bojanowski P., Douze M., Jégou H., Mikolov T.** Fasttext. zip: Compressing text classification models. 2016. arXiv preprint arXiv:1612.03651.
6. **Chowdhury G. G**. Natural language processing, *Annual review of information science and technology,* 2003, vol. 37, no. 1, pp. 51—89.
7. **Devlin J., Chang M. W., Lee K., Toutanova K.** Bert: Pre-training of deep bidirectional transformers for language understanding. 2018 Oct 11. arXiv preprint arXiv:1810.04805.
8. **Ethayarajh K.** How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 2019. arXiv preprint arXiv:1909.00512.
9. **Howard J., Ruder S.** Universal language model fine-tuning for text classification. 2018. arXiv preprint arXiv:1801.06146.
10. **Génova G., Fuentes J. M., Llorens J., Hurtado O., Moreno V.** A framework to measure and improve the quality of textual requirements, *Requirements Engineering,* 2013, vol. 18, iss. 1, pp. 25—41, DOI: 10.1007/s00766-011-0134-z.
11. **INCOSE**, Guide for writing requirements INCOSE TP-2010-006-02, USA, San Diego, International Council on Systems Engineering, 2015, 73 p.
12. **Kotsiantis S. B., Zaharakis I., Pintelas P.** Supervised machine learning: A review of classification techniques, *Emerging artificial intelligence applications in computer engineering,* 2007, vol. 160, no. 1, pp. 3—24.
13. **Ke G., Meng Q., Finley T.** et al. Lightgbm: A highly efficient gradient boosting decision tree, *Advances in neural information processing systems* 30, 2017, pp. 3146—3154.
14. **Nasteski V.** An overview of the supervised machine learning methods, *Horizons*, 2017, vol. 4, pp. 51—62.
15. **Hossin M., Sulaiman M. N.** A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 2015, vol. 5, no. 2, pp. 1—11. DOI: 10.5121/ijdkp.2015.5201.

ИНФОРМАЦИЯ

4—6 октября 2022 г. в Санкт-Петербурге на базе АО "Концерн" ЦНИИ "Электроприбор" состоится 15-я мультиконференция по проблемам управления (МКПУ-2022)

Мультиконференция включает пять локальных конференций:

- **XXXIII конференция памяти выдающегося конструктора гироскопических приборов Н. Н. Острякова**
- **Конференция "Информационные технологии в управлении" (ИТУ-2022)**
- **Конференция "Математическая теория управления и ее приложения" (МТУиП-2022)**
- **Конференция "Управление в аэрокосмических системах" имени академика Е. А. Микрина (УАКС-2022)**
- **Конференция "Управление в морских системах" (УМС-2022)**

В рамках мультиконференции пройдет Семинар по закрытой тематике

Информация для связи:

ГНЦ РФ АО "Концерн "ЦНИИ "Электроприбор",
Тел.: +7 (812) 499-82-10 — Истомина Елена Анатольевна,
+7 (812) 499 82 67 — Тарановский Дмитрий Олегович,
E-mail: icins@eprib.ru