DOI: 10.17587/prin.13.187-199

Applying Unsupervised Machine Learning Algorithms to Ensure Requirements Consistency[1]

K. I. Gaydamaka, k.gaydamaka@gmail.com, Department of Systems Engineering, "MIREA — Russian Technological University", Moscow, 119454, Russian Federation,
A. D. Belonogova, alena.belonogova@yandex.ru, National Research Nuclear University "MEPhI", Moscow, 115409, Russian Federation

Corresponding author:
Gaydamaka Kirill I, Postgraduate Student, 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 March 05, 2022

The article is devoted to the problem of ensuring the quality of requirements for complex technical systems. The purpose of this article is to apply unsupervised machine learning techniques to test a set of requirements for consistency. It is assumed that the clustering of requirements will allow us to determine the requirements that are closest in meaning to the given one, and this may indicate a possible contradiction and require additional check of potentially conflicting requirements. The article discusses a comparison of clustering methods such as k-means, agglomerative hierarchical clustering, DBSCAN, EM-algorithm, as well as methods for converting sentences into numeric vectors TF-IDF, doc2vec, BERT. BERT and k-means showed the best result — a cluster that included only conflicting requirements.

Keywords: *natural language processing, requirements clustering, requirements management, requirements engineering, requirements consistency*

For citation:

Gaydamaka K. I., **Belonogova A. D.** Applying Unsupervised Machine Learning Algorithms to Ensure Requirements Consistency, *Programmnaya Ingeneria*, 2022, vol. 13, no. 4, pp. 187—199.

DOI: 10.17587/prin.13.187-199

УДК 004.852

К. И. Гайдамака, аспирант, k.gaydamaka@gmail.com, РТУ МИРЭА,
А. Д. Белоногова, магистр, alena.belonogova@yandex.ru,
Национальный исследовательский ядерный университет "МИФИ"

Применение методов машинного обучения без учителя для обеспечения непротиворечивости требований

Статья посвящена проблеме обеспечения качества требований к сложным техническим системам. Цель этой статьи — применить методы машинного обучения без учителя, чтобы проверить набор требований на непротиворечивость. Предполагается, что кластеризация требований позволит определить наиболее близкие по смыслу к заданному требования, а это может указывать на возможное противоречие и потребовать дополнительной проверки потенциально конфликтующих требований. Рассматривается сравнение таких методов кластеризации, как k-means, агломеративная иерархическая кластеризация, DBSCAN, EM-алгоритм, а также методы преобразования предложений в числовые векторы TF-IDF, doc2vec, BERT. Наилучшие результаты показал BERT в тандеме с k-means, что позволило получить кластер, в который включены только конфликтующие требования.

Ключевые слова: *обработка естественного языка, обучение без учителя, кластеризация, управление требованиями, инженерия требований, непротиворечивость требований*

[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.

Introduction

Requirements make it possible to determine what stakeholders want to get from the system being created and what properties the system should have to satisfy their needs [1]. They are necessary so that the project team has a clear and consistent idea of what system will need to be developed, what functionality this system should have and what problem the end user should solve. One of the key and important stages of work on a project is the stage of developing system requirements. It is at this stage that it becomes possible to designate the boundaries of the problem area, then to select from it a specific list of problems, and when formulating requirements, compare each problem to some specific solution. The result of such work will be a description of the concept of a system that will be able to completely solve all the client's problems, and therefore will be economically profitable [2].

The development of high-quality system requirements plays a decisive, key role in the process of working on a project, regardless of its scale, be it the design of a nuclear power plant or the creation of a portable audio player. The timing of implementation, the cost of development and other indicators of the project's effectiveness directly depend on how accurately and correctly the system requirements describe the current reality, determine the key "problem" of the client and, in a language that is simple and understandable to all members of the project team, describe the criteria that the system must satisfy in the final the end result [3]. Cases where insufficient resources are allocated to requirements development certainly entail risks related to the earliest stages of the project work life cycle. And as you know, risks of this nature can cause the most significant damage to the cost and timing of the project. This is why it is necessary to pay special attention to requirements engineering in design practices.

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. And since the development of requirements is entirely on the shoulders of analysts, it would be logical to assume that it cannot but depend on the human factor: inattention, negligence, haste. But in cases where huge sums of money are at stake, and due to the imperfection of the human resource, the project executor may suffer significant losses, it is necessary to minimize the human factor.

An excellent example of the situation described above would be a possible conflict of system requirements with each other. When several thousand requirements describe a large-scale system, and a staff of analysts is working on their development, it is easy to overlook such a case, as if requirement #78 said "The kettle must boil in at least 50 seconds." not less than 60 s. How easy it is to make such an annoying mistake, it is just as difficult for a person to find it, and only an automated intelligent tool can help improve the situation. This is precisely the purpose of this work — to provide an automated consistency check of system requirements.

In particular, it is assumed that the developed algorithms will make it possible to determine the requirements that are closest in semantic meaning to a certain given one, after which the algorithm will issue a recommendation to re-check the requirements from the found group for contradictions with each other. Undoubtedly, one cannot expect that the system will unambiguously and accurately determine the presence or absence of the aforementioned problem, however, one can definitely count on the fact that it will be able to competently separate the requirements by meaning and divide them into a large number, depending on the number of requirements, semantic groups — clusters. And taking into account the amended condition, the size of each cluster will not exceed two to four elements, and the presence of requirements in one cluster can potentially signal the problem of inconsistency. In this case, the analyst is left with checking several proposals, instead of a full-fledged revision of the requirements.

1. Writing the requirements

Regardless of what area the system being developed belongs to, what scale it is, the requirements should be easy to read and easy to work with.

For easy readability, it is assumed that the wording of the requirement should follow some special pattern, such that it will be convenient for the user to perceive it [4]. For example, [User Category] must have the [Capability] feature. The analyst's tasks are:

- a competent choice of a requirement template;
- substitution of values of quantities in the corresponding gaps in the requirement template.

Using patterns allows you to generate requirements at any time without spending more resources on it than possible, as well as:

- simplifies the process of changing several requirements of the same type — it is enough to change the parent template;
- simplifies the process of searching for requirements, for example, by filtering by a specific value of the initiating event;
- allow you to encapsulate confidential information, since it can be stored in a closed loop, and the process of forming a request will only have an interface for obtaining a request template, substituting secret data in the appropriate sections.

In the case of light work, the requirements should provide a range of possibilities. Below is a sample list.

- The ability to distinguish between requirements by unique identifiers.
- The ability to classify requirements for a certain set of parameters.
- Ability to monitor changes in the statuses of requirements.
- The ability to evaluate and analyze requirements as part of a whole (requirements specification).
- The ability to search the specification of requirements in order to find a specific requirement based on the context.
- The ability to establish relationships between requirements.

Based on this, it becomes obvious that it would be bad form to match the unique identifier of the requirement with the content of its content. Therefore, you should use meta information for each unit of the requirements

specification, and put some defining requirements in the meta information. Requirement attributes are called these values. The use of attributes will allow, firstly, to distinguish requirements as separate objects with unique identification numbers, which further implies the possibility of establishing links between objects. Second, the presence of meta information for each requirement allows, without reading into the text of the requirement itself, to obtain important characteristics just by simply "skimming" a glance at the specification. Third, meta information denies the overload of the requirement text — there is no need to indicate the priority, the verification method in the text itself, because such data can be taken out into meta information. And finally, the last, search capabilities are multiplied many times, if the search can be carried out not only in the text of the requirement, but also in the fields of meta information.

However, these tips are not enough to ensure the quality of the requirements. The quality of the requirements can be judged by the presence of the following properties:

- uniqueness — each requirement must have a unique identifier to distinguish it from others;
- feasibility — there must be an opportunity to implement the requirement within a specified time frame and with a specified budget;
- legality — the requirement must not contradict applicable laws;
- clarity — there should be no ambiguity in the requirement;
- accuracy — the requirement should be concise;
- verifiability — a strategy should be in place to verify that the requirement has been met;
- abstractness — the requirement should not define the implementation.

With regard to the dataset, there are the following characteristics:

- completeness — the requirements cover the entire functionality of the system;
- consistency — there are no conflicting requirements;
- no redundancy — there are no repetitions in the requirements;
- modularity — requirements that are similar in meaning are in the same category;
- structuredness — there is a clear structure in the requirements specification;
- satisfaction — all requirements marked "satisfied by" refer to existing requirements;
- testability — requirements are covered by tests at the desired level.

Inconsistency of a requirement or a set of requirements even with one criterion indicates that the specification is of poor quality, and that problems are likely to arise during the further design and implementation of the system. Therefore, the only solution is the timely correction of the error at the earliest stages of the project development. First of all, in order to make changes to the incorrect wording, it is necessary to detect the problem. However, when the requirements specification is calculated, in the case of large-scale projects with tens of thousands of requirements, the problem lies precisely in finding an error (even though it may not exist at all). In this case, an expert team is involved, which re-checks the wording of the requirements. But if the discrepancy between a separate requirement and a separate criterion is checked rather quickly (although a full check still takes a long time), then checking a set of requirements can take up a significant amount of resources, even without taking into account the human factor (i. e., that the error will not be detected even after repeated verification). Therefore, in this case, it makes sense to create an intelligent system that will help the team of systems engineers in requirements management processes.

In this work we will focus on one criterion. As part of this work, it is necessary to create a system that can read a set of requirements and check them for consistency. In other words, this system will determine the closest in meaning requirements to a user-specified requirement and issue a recommendation to check three or four proposals instead of the entire specification.

2. Description of the approach to identifying conflicting requirements

In the previous section, we identified the core of requirements engineering and requirements management discipline — the resource-intensive process of checking a set of requirements for inconsistency, including human factors. But it is necessary to assume how the invited expert would solve this problem? Of course, the easiest way is to consistently go through the list of requirements and, making pairs "each requirement with each requirement", check them for contradiction. But, obviously, this process is incredibly time consuming. Assuming that it will take ≈ 20 s to compose and verify one pair of requirements, how long will it take to verify a requirement specification consisting of 500 elements? Applying the formula (1) combinations of 2 elements out of 500 from combinatorics, we carry out elementary calculations:

$$C_n^k = \frac{n!}{k!(n-k!)} \to C_{500}^2 = \frac{500!}{2! \times 498!} = \\ = 124\,750 \text{ pairs} \to 124\,750 \cdot 20 \approx 693 \text{ hours}. \quad (1)$$

Obviously, this is not possible to spend about a month without interruption on such a check "head-on". Most likely, the requirements specialist would try to divide the specification into logical subgroups if the requirements were in the same subgroup — this means that they are similar in meaning. And then, instead of checking the consistency of the combination of all requirements with all, he would perform a similar task within one small structure. While we do not take into account the process of dividing the requirements into groups, we will assume that this task has already been completed: there are 50 groups of 10 requirements each. Then our formula for calculating the amount of time spent on the expert assessment of consistency will look like this:

$$C_n^k = \frac{n!}{k!(n-k!)} \to C_{10}^2 = \frac{10!}{2! \cdot 8!} = \\ = 45 \text{ pairs} \to (45 \times 20) \times 50 = 12.5 \text{ hours}. \quad (2)$$

Thus, the time was reduced by an order of magnitude, however, checking 50 groups of 15 minutes each, which

is still quite a long time, but at least once this task can be completed.

In addition, it is possible to radically change the approach to solving this problem — *not to check all the requirements in the aggregate, but to check the contradiction of the new requirement of the existing specification*, and not even the entire specification — for each new requirement the closest semantic group will be selected, and only inside her. At the beginning of the development of the requirements specification, the number of groups and the number of requirements within the groups will be very small, and by the time the specification has grown enough to have 10 requirements in each group, the old requirements will be tested long ago. This approach is carried out in approximately the same time frame as the previous one.

But you can go even further in thinking — if you introduce a *metric* according to which it will be possible to determine the distance between requirements, and those that are closer in semantics will be at a shorter distance than more distant ones. Then it is necessary to enter some *threshold value* and establish that if the distance between the requirements is below this value, then this means that the requirements contradict each other. In this case, it is possible to carry out a full check of all requirements. If you measure the distances within semantic groups, then you will need to personally check only those formulations of requirements, the distance between which is below a certain threshold value.

And here we come to the most difficult question: *how exactly to divide requirements into groups?* Different requirements can refer to different levels of decomposition, describe different functional features, and it is not always clear where one semantic group ends and another begins. If asked to divide requirements into groups, an examiner will most likely separate each N requirements of the specification into a separate structure, assuming that the specification is structured and sequential. However, conflicting requirements can be at different ends of the specification, and splitting them up in this way will not only get the project team closer to solving the problem, but rather wasteful validation process will take up valuable time. As for the idea of introducing a metric for requirements, this defies commentary, because the expert assessment in this case will be a subjective value, as well as the division into semantic groups is also very subjective.

Therefore, we need a tool devoid of any subjective view. This tool is an electronic computer, all value judgments of which have a mathematical basis.

And so, getting close to the value judgments made by the computer, we begin to dive into machine learning.

Machine learning (ML) addresses the question of how to build computers that improve automatically through experience [5]. In another way, it can be formulated like this — it is a process of searching for patterns in a large amount of information and making the best decision without human participation.

Machine learning is used in many areas of daily life. Smart "news feeds", contextual advertising, face recognition systems — all these are the results of ML algorithms. In addition, other experimental solutions are already being actively implemented, for example, the construction of a treatment regimen for a patient without human intervention. Definitely, in the future, the field of application of machine learning will only increase.

There are three main areas of ML [6]. The first thing we'll talk about is supervised learning. The tasks of this direction are reduced to approximately one thing: it is necessary to analyze the incoming labeled dataset and test some hypothesis. The second type of machine learning is deep machine learning — it can only work with what's called big data.

But we will be more interested in the third direction of machine learning — ML without a teacher. Unsupervised machine learning operates only with objects — class labels, one might say, are not available. And thus the algorithm must draw its own conclusions from the input datasets, without having any examples, in order to make decisions by analogy. Here are the tasks that ML solves without a teacher:

- clustering task;
- the task of finding association rules;
- anomaly detection task;
- dimensionality reduction problem;
- data visualization task.

The clustering task means dividing the original dataset into separate groups, called clusters, so that the objects of one cluster are similar, and the objects of neighboring clusters are different. The task of searching for association rules implies that the data is presented in the form of feature descriptions, and it is necessary from the entire array of features to find such feature values that are most often found in feature descriptions of objects. The task of finding anomalies is that the algorithm must determine an unusual set of data among all, that is, to detect that some objects behave in a fundamentally different way than expected from it. The problem of dimensionality reduction is posed when each object in the initial data has a large number of characteristics that determine it, and it is necessary to select from all of them only the most informative parameters that determine the behavior and, finally, the task of data visualization is used when it is necessary to somehow represent multidimensional quantities — in this case, objects from N-dimensional space are translated into one- two- or three-dimensional, after which they are reflected on a flat graph or in space.

Now let's go back to where we started — you need a tool that is devoid of a subjective view or is equally subjective in all cases, which will allow you to divide the specification of requirements into separate subgroups, and such that the requirements in different subgroups will differ in meaning, and in the same — be similar. We got the definition of clustering, in other words, according to our initial reasoning, it is necessary to cluster the requirements. And the unsupervised machine learning model just knows how to solve the clustering problem.

Thus, to summarize: to determine whether a new requirement contradicts the specification, it is necessary to cluster the specification and check the contradiction of the new requirement with those that are in the same cluster with it. In order to check the entire specification for consistency, it is necessary to cluster the specification, introduce the metric of the distance between the requirements within the cluster, and set the limit value at

which the requirements should be located without being conflicting, check all cases when the distance between the requirements is less than a given value. To do this, you need to understand how clustering works, which will be covered in the next section.

3. Methods for solving the clustering problem

When solving the clustering problem, we have N objects — $x_1 ... x_N$, each of which must belong to some class, as well as class labels $y_1 ... y_M$, $M \leqslant N$. For example, objects can be bicycles, or rather, their parametric description — weight, length, number of speeds, and class labels — the type of bicycle — children, sports, walking. Objects can be people of a specific height and weight, and class labels can be gender (male or female). Also, the objects can be requirements, and labels — the number of the semantic group, which the requirement belongs to. The task of clustering is to correctly place labels on objects in such a way that similar objects to each other fall into the same class, and objects that are fundamentally different — into different classes.

You can measure the quality of the clustering operation performed. We will definitely want the intra-cluster distance to be as small as possible and the distance between the clusters as large as possible. In mathematical terms, the formula is as follows: let F_0 be the average intra-cluster distance, and F_1 is the average distance between clusters. Then these values are calculated according to the formulas (3) and (4), and it is necessary to minimize their quotient, as shown in the formula (5):

$$F_0 = \frac{\sum_{i<j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i<j} [y_i = y_j]} \to \min. \quad (3)$$

$$F_1 = \frac{\sum_{i<j} [y_i \neq y_j] \rho(x_i, x_j)}{\sum_{i<j} [y_i \neq y_j]} \to \max. \quad (4)$$

$$\frac{F_0}{F_1} \to \min. \quad (5)$$

Clusters can be nested within each other. Depending on the level of decomposition, two objects can be located both in the same cluster or in different ones. With hard clustering, one object can be assigned to only one cluster; with soft clustering, it can be assigned to several classes with an indication of the weight. Clusters come in many different shapes, and fig. 1 shows an illustration of six different cluster views. Obviously, some methods do well with some forms, others with

Fig. 1. Various forms of clusters

others. And speaking of clustering methods, now we will analyze the main models.

The easiest way of clustering is k-means [7]. Objects that we clusterize are points in N-dimensional space. You must first select a number k — the number of clusters. It is assumed that it is known. Then all of a set of points you need to randomly select k pieces — this will be the centers of clusters. Then the distance of each point are measured for each center point is determined and in that cluster, the center of which it is closest. His new center, after which the operation is repeated as long as the change in position of the center of the cluster will not be substantially larger (smaller than a certain threshold value) for each of the resulting cluster is determined — then the cluster centers are recalculated. This method of k-means behalf Ball Hall. In the implementation of McKean cluster centers are recalculated whenever the object. When working with large data, not considering the arithmetic mean of all the objects of the cluster, it is taken from the random sample and the center is considered to be on it — it is an implementation of the Mini Batch. There is another implementation of k-means — the k-means++. In her initial approach is not chosen at random, and with a uniform distribution. And each successive approximation center of the cluster is set to be the most plausible in this situation. Fig. 2 (see the 2nd side of cover) shows an example of clustering added using this algorithm.

The result of the k-means method: in order to initiate the k-means object, you need to specify the number of clusters. The Mini Batch implementation allows you to work on a very large sample. The distance between the clusters is Euclidean. The k-means method is best used when the goal of the experiment is to identify clusters of approximately the same size.

The next clustering method is the EM (expectation—maximization) algorithm [8]. Let there be a sample of some random variable, and our task is to cluster the objects of this sample. The essence of the algorithm is in two steps — E and M. At step E, we need to define some hidden variables. At step M, depending on the values of the hidden variables, we calculate the probabilities of determining a particular object to a particular cluster and recalculate the hidden variables. Let our random variable be uniformly distributed, and then the hidden variables that we will determine at step E will be the mathematical expectation μ the value of the variance σ. Then, at step E, substituting certain μ and σ, we obtain the value of the probability density of attributing one or another object (x_i) to one or another cluster (j):

$$p_j(x_i) = \frac{1}{\sigma_i} \exp\left(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2} \right). \quad (6)$$

Then, applying the Bayes formula, the probabilities of determining the i-th object to the j-th cluster are calculated:

$$g_{ji} = P(j \mid x_i) = \frac{w_j p_j(x_i)}{\sum_{k=1}^{K} w_k p_k(x_i)}. \quad (7)$$

At the second step (M-step), the likelihood maximization problem with fixed values of $P(j \mid x_i)$ is solved. If we equate the derivatives with respect to the parameters to zero, we obtain the expressions:

$$w_j = \frac{1}{N}\sum_{i=1}^{N} g_{ji},\ \theta_j = \arg\max_{\theta} \sum_{i=1}^{N} g_{ji} \ln \phi(\theta;\ x). \qquad (8)$$

At the same step, the hidden values are recalculated and the transition to step E is carried out. The cycle continues until convergence occurs (i. e., changes in the probabilities of assigning objects to different classes do not exceed some value specified by the user).

The clustering problem using the EM-algorithm is the problem of dividing a mixture of distributions: in which it is necessary to estimate the weights w_j and parameters θ_j of individual components. Figure 3 (see the 2nd side of cover) shows a visual reproduction of the application of the algorithm that was described above.

Summing up the results of EM clustering, to create an instance of a class, you need to specify the number of components. A model on a small sample can be retrained, i. e. get too close to a specific sample, and at the same time show a low result on test data, therefore, it is preferable to use it on a large sample. This model restores the distribution density; the resulting clusters are most likely to be convex. The distance between clusters is calculated in Euclidean metric.

The next technique for clustering is agglomerative hierarchical clustering [9]. First of all, let's define what is hierarchical clustering? This is clustering, in which clusters can be nested within each other. There are two fundamentally different methods for identifying such clusters:

- agglomerative — at the first stage, each object is placed into its own cluster, and at each next step, close clusters are combined into one.
- divisional — at the first stage, all objects are placed in one cluster, and at each next step, the cluster is split into smaller clusters.

The most common method is the first, and therefore in this work we will talk about agglomerative hierarchical clustering. This algorithm is preferable to use when we are talking about a strictly specified number of clusters — in this case, it is enough to wait until the objects are combined into a set of clusters, and the cardinality of the set will coincide with a fixed value.

In fact, this algorithm appears to be intuitive. The complexity can arise only when we think about how to calculate the distance between clusters? There are several possible ways. The distance between two clusters is the distance between the nearest neighbors of these clusters. Formalizing the above, we get:

$$R^{\text{Б}}(W,\ S) = \min_{w \in W,\ s \in S} \rho(w,\ s). \qquad (9)$$

Similarly, the distance between two clusters can be defined as the distance between the most distant neighbors of these clusters:

$$R^{\text{Д}}(W,\ S) = \max_{w \in W,\ s \in S} \rho(w,\ s). \qquad (10)$$

You can calculate the average distance between all objects of one and the other clusters:

$$R^{\text{С}}(W,\ S) = \frac{1}{|W||S|}\sum_{w \in W}\sum_{s \in S} \rho(w,\ s). \qquad (11)$$

Another way is to calculate the centers of the clusters and consider the distance between them as the distance between the clusters:

$$R^{\text{Ц}}(W,\ S) = \rho^2\left(\sum_{w \in W} \frac{w}{|W|},\ \sum_{s \in S} \frac{s}{|S|}\right). \qquad (12)$$

Finishing with the problem of measuring the distance between clusters, we propose another possible measure — the Ward distance:

$$R^{\text{У}}(W,\ S) = \frac{|S||W|}{|S| + |W|}\rho^2\left(\sum_{w \in W} \frac{w}{|W|},\ \sum_{s \in S} \frac{s}{|S|}\right). \qquad (13)$$

Figure 4 shows a step-by-step view of the agglomerative hierarchical clustering process.

Summing up the discussion of agglomerative hierarchical clustering, to initialize the model, it is necessary to set the number of clusters, the method for calculating the distance between clusters from those described above, and the metric. This method shows itself well in the case of a large number of objects, and in the case of a large number of clusters — in fact, in this situation the model demonstrates the best results, and any metric can be set.

We now turn to the final clustering model, the density-based model DBSCAN [10]. Such models work on the assumption that objects are a set of points distributed with certain densities, and in the vicinity of some points there are other points. Let there be some point (object) consider its neighborhood of radius R. If there are N or more other point objects in the specified neighborhood, then such a point is the main one. If in the vicinity of points there are less than N, but there is at least one main point, then it will be called a boundary point. In all other cases, it is noise.

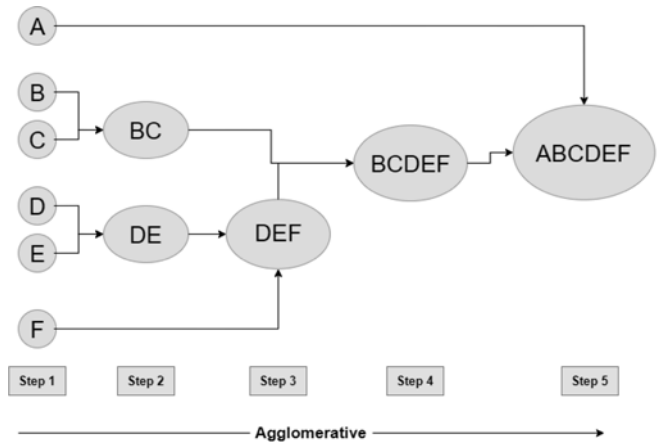

Fig. 4. Agglomerative clustering

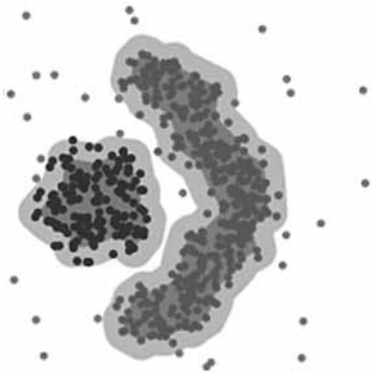

Fig. 5. DBSCAN Clustering

The DBSCAN algorithm is as follows.

- It is necessary to assign each point to one of the classes (main, borderline, noise).
- Connect the points at a distance L to each other — the result of the work is a graph.
- Combine all main points connected by a line into one cluster — in other words, select the components of the graph's connectivity.
- Assign the remaining border points to the corresponding clusters.

In practice, DBSCAN shows high results on complex shapes of clusters, it perfectly detects noises that pretty much spoil other clustering algorithms. However, when dealing with simple cluster shapes that are close to each other, DBSCAN can be wrong. Figure 5 shows the result of this algorithm.

Summing up the conversation about DBCAN, to initialize the model, it is necessary to set the radius of the neighborhood in which to search for nearby points, and the number of neighbors that the point must have in order for it to become the main one. The algorithm shows good results on a large number of objects and a small number of clusters and does an excellent job in identifying non-trivial forms. The distance between clusters is calculated in Euclidean metric.

4. Methods for vector representation of sentences

Let's start with the simplest method, the simple cosine distance method [11]. The first step of the algorithm is to tokenize sentences into words. The next step is lemmatization, i. e. reducing all words to their normal forms and excluding everything except words. The third step of the algorithm is to determine the set of words for all these sentences. Let us define a vector space **V** of dimension n, where n is the cardinality of the set, and assign to each word from the set some vector in **V** so that the obtained vectors constitute a system of pairwise orthogonal vectors. Thus, we get that the j-th coordinate of the i-th word will be equal to:

$$a_{ij} = \begin{Bmatrix} 1 | i = j \\ 0 | i \neq j \end{Bmatrix}. \qquad (14)$$

The fourth step of the algorithm is to determine for each sentence of our text its vector in the space **V**. This procedure can be described in simple words: we sequentially consider each word of the "bag of words" [12], if the i-th word of the dictionary is present in the sentence, then its i-th the coordinate of the vector of the corresponding sentence takes the value of one, and otherwise — zero. This procedure can be clearly seen at table 1.

Table 1

Definition of vectors from word space

Sentence ID	Value	Division	Timer	Shall	Be	Second	Error	Measure	Time
1	1	1	1	1	1	1	0	0	0
2	0	0	1	1	1	1	1	0	0
3	0	0	1	1	0	0	0	1	1

The final stage of the algorithm is to select the sentence for which we want to determine the closest sentence in meaning, determine the cosine distances between the vector corresponding to the target sentence and all other vectors. The minimum cosine distance means the greatest semantic similarity. According to the formula (15) the order of calculating the cosine distances between vectors is determined:

$$\rho_{\cos}(x,\ y) = \arccos\left(\frac{\langle x,\ y\rangle}{\|x\| \cdot \|y\|}\right) = \\ = \arccos\left(\frac{\sum\limits_{i=1}^{d} x_i y_i}{\sqrt{\sum\limits_{i=1}^{d} x_i^2} \cdot \sqrt{\sum\limits_{i=1}^{d} y_i^2}}\right). \qquad (15)$$

Applying the resulting formula to the vectors above, we get that $\rho_{\cos}(a_1,\ a_2) \approx 0.73$, $\rho_{\cos}(a_1,\ a_3) = 0.4$, $\rho_{\cos}(a_2,\ a_3) \approx 0.45$. The algorithm determined, as expected, the first and second sentences as the closest pair, and the first and third sentences as the farthest. But this trivial approach only works with simple examples like the one above. With regard to the practice of developing requirements, in the specification it is quite likely that requirements of the form: "The walls of the building should be painted blue", "The ceiling of the building should be painted white". Obviously, the algorithm described above will take into account the almost identical filling of both sentences with words, and will issue a recommendation to check these requirements for possible inconsistency. However, there is no contradiction, because the key and most important detail in this proposal is not the description of the function (should be painted), but the object that this function is aimed at – the walls or ceiling. Therefore, the transformation of sentences into numerical vectors based only on the presence or absence of certain words is obviously fraught with problems. Moreover, this algorithm in no way took into account the frequency of occurrence of words, but took into account the words "common" for all requirements, which should not be.

Regarding the last remark, of course, there is a high-quality add-on for this algorithm. It's about TF-IDF Vectorizer.

TF-IDF (Term Frequency Inverse Document Frequency) [13] is a very common sentence-to-numeric vector algorithm used in unsupervised machine learning. TF-IDF is an algorithm that determines a measure of the originality of a word by comparing the number of times a word appears in a document with the number of documents in which that word appears. The calculation of the originality of a word is performed according to the following formula:

$$TF\text{-}IDF = TF(t,\ d) \times IDF(t) = \\ = TF(t,\ d) \times \log \frac{1+n}{1+df(d,\ t)} + 1, \quad (16)$$

where $TF(t,\ d)$ is the frequency of the word, the number of times the word t occurs in the document d, is the reciprocal frequency of the document, calculated according to n — the number of documents, and $df(d,\ t)$ — the number of documents in which the given word occurs.

The algorithm is as follows — at the first step, you also need to create a dictionary based on sentences (documents), having previously reduced them to normal form. We will consecutively number all the words that occur in sentences. Stop words (for example, for the Russian language these are conjunctions, prepositions) will not be counted and numbered, since they do not represent any information. The second step is to transform the sentence space into a vector space. We define the following function $TF(t,\ d)$:

$$TF(t,\ d) = \sum_{x \in d} FR(x, t);\ FR(x,\ t) = \begin{cases} 1, x = t \\ 0, x \neq t \end{cases}. \quad (17)$$

As a result, sentences $d_1 \ldots d_m$ in the word space $t_1 \ldots t_n$ will have the following form:

$$\vec{v}_{d_i} = \left(TF(t_1,\ d_i),\ TF(t_2,\ d_i),\ \ldots,\ TF(t_n,\ d_i)\right). \quad (18)$$

Combining the vectors of the sentence space into a matrix, we get a matrix $TF_{m \times n}$. The next step is to calculate IDF using formula (17), as a result of which the resulting vector of dimension n is reduced to the form of a diagonal matrix IDF of size $n \times n$. At the final stage, the matrices TF and IDF are multiplied, and the obtained normalized result is a matrix, along the rows of which the sentence vectors are located in the numerical space. This result is somewhat more meaningful than the method of simple cosine distances, it takes into account the frequency of words both within a sentence, as in other sentences of the text, but the algorithm still depends on the presence of a particular word in the sample and does not take into account different formulations of the same essence.

The latest Google development is the Word2Vec algorithm [14], which transforms words into the space of numeric vectors, taking into account the semantics of words. There are two fundamentally different approaches to Word2Vec — CBOW and skip-gram. Let's take a look at each of these approaches in turn.

The CBOW (Continuous bag of words) method allows you to predict words based on context. Suppose you have an input sentence "It is raining outside the window." The CBOW method will read the words "for", "window", "rain" and will try to predict the word "pouring". Words in Word2Vec are represented as N-dimensional vectors. A pretrained neural network is used to transform a word as a set of letters into an N-dimensional vector. With each prediction of a word from the context, the neural network is retrained and determines the value of the new N-dimensional vector as a new word.

The skip-gram method, in another case, allows one to predict the context one word at a time. For example, using the word "pouring" to restore the words "for", "window", "rain". This method is somewhat slower, but works better with words that are infrequent. It also uses a neural network.

The result of the work of any of the Word2Vec algorithms are N-dimensional vectors that define the words of the sentence. The advantage of Word2Vec over the TF-IDF algorithm described above is that TF-IDF vectors have a dimension that matches the cardinality of a dictionary per document — and this is an order of more than 100,000, while the dimension of Word2Vec vectors ≈ 300. Therefore, the resulting vectors can be used in supervised machine learning methods, for example, in the classification problem, if we consider each coordinate of the vectors as a feature value. But the main advantage of Word2Vec is the restored semantic relations between the words of the dictionary. The resulting numeric representations of words allow relational operations on words. The most striking and recognizable example of Word2Vec sounds like this: the difference between the words "king" and "queen" is equal to the difference between the words "man" and "woman", i. e. "King — queen = man — woman".

However, when working with requirements, it is not enough for us to be able to transform individual words into vectors — it is necessary to do the same with whole sentences. The first intuitive and method that comes to mind will be averaging all the word vectors in a sentence, but this approach does not take into account the change in word order in any way, so let's delve deeper into the search for a technology that extends Word2Vec to the entire document — and such a technology exists and is called Doc2Vec.

According to [15], the Doc2Vec approach can be described as follows — each sentence is transformed into a numeric vector, which is a column in the matrix D, each word is also transformed into a numeric vector according to the well-known Word2Vec algorithm, which is a column in the matrix $\mathbf{W}$. Thus, in addition to use only vectors of words to predict another word, we additionally introduce a sentence vector that is unique for each sentence. Thus, the result of training the neural network is the sentence vector we are looking for.

As in the case of Word2Vec, there are two approaches to defining the sentence vector — DM (distributed memory) and DBOW (distributed bag of words). DM allows you to define a word from a context, DBOW — a context from a sentence vector.

Summing up the consideration of this model, we can say with confidence that doc2vec is a new step towards

the development of technology for transforming sentences into numerical vectors. However, it also has a number of disadvantages, first of all, it is the dependence on the independent learning of the neural network, and this requires rather large data. From the articles on the implementation of doc2vec, it was found that if you train doc2vec on data of 100,000 sentences, then the accuracy of the results of the regression algorithms, classification, using the vector coordinates obtained as a result of the doc2vec algorithm as a feature, is only 74...76 %. Certainly, it cannot be argued that if the supervised machine learning models show such a mediocre result, then the clustering of the obtained vectors will be insignificantly better, but this makes one think. And returning to the context of the application of algorithms — the specification of requirements — you involuntarily ask yourself the question "will there be 100,000 proposals in the specification of the requirements of any industrial project?" Of course not, not in all. But definitely, the assumption that the doc2vec model will perform poorly on small amounts of data should be tested.

The final model to be discussed in this paper is BERT [16]. The BERT model was announced and made public in 2018. In addition to the fact that the algorithm demonstrated the highest results of work, the BERT developers provided already trained data models on large datasets, and thus relieved potential system developers who decided to use BERT from training problems. It is enough to import the finished model into the code, after which it is possible to start development.

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 community as learning with partial involvement of a teacher, ELMo models (Embeddings from Language Models), ULMFiT (Universal Language Model Fine-Tuning), 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.

Let's note some more unique features of BERT. BERT is not a single directional model, it means that it does not perceive words sequentially from left to right, for example. Such a method definitely allows one to take into account the context of words without reference to two neighbors. Obviously, the most difficult stage in the life of a BERT is the prediction stage, which corresponds to the results of the development of the encoder propagation layer. Whereas conventional models predict words from left to right, BERT avoids this solution in the following way: before reading a list of words, a BERT component called a marker replaces 15 % of each line, thus masking the terms. After that, BERT tries to guess the changed words in accordance with the context in which the marker did not work. To predict a specific word, you first need to introduce a word classification tool at the encoder output and calculate the probability of the presence of each word using softmax.

By receiving different words from the same sentence as input, BERT tries to predict whether the second word is a logical continuation of the first, assuming that half of all input is pairs and the remaining half of the words are random. In order for the model to understand the difference between the two words during training, the CLS indicator is inserted at the beginning of each word, and the SEP indicator at the end.

To summarize the description of BERT, it is an extremely efficient model for NLP problems and extremely difficult to understand. The presence of an already trained neural network of several hundred hidden levels allows you to obtain the highest results in solving the problem, without spending computer resources and time resources. And multilingualism allows you to work with the semantics of sentences in a variety of languages.

5. Results

This chapter presents the results of applying the algorithms described above to test data. As test data, a specification of requirements from 282 requirements in Russian was used, and several of them were artificially added to create inconsistencies that will need to be found by our algorithms. Therefore, two groups of contradictions were created, each with three requirements. The first group contradicts in terms of the timer division value, and the second — in terms of errors:

R-0224: Цена деления таймера должна составлять 1 минуту

R-0278: Цена деления шкалы таймера 1 мин

R-0282: Цена деления шкалы таймера 2 мин

R-0238: Погрешность таймера должна составлять 0,5 мин.

R-0221: Погрешность таймера должна составлять 1 мин.

R-0277: Таймер должен производить отсчет времени с погрешностью не более 0.5 мин

As can be seen from these contradictions, a couple of sentences in theory should be defined very simply — they simply change the number corresponding to either the division price or the error. For each group, a contradiction was introduced with a reformulation — if the system is able to identify such contradictions too, this can be regarded as a success.

First, let's try to compare clustering models with each other. Let us fix the TF-IDF algorithm as an algorithm for obtaining vectors from sentences and successively try to determine the cluster for the requirements R-0282,

	ID	Requirement		ID	Requirement
221	R-0224	Цена деления таймера должна составлять 1 минуту	209	R-0212	Рабочий угол таймера должен составлять 360 гра...
275	R-0278	Цена деления шкалы таймера 1 мин	218	R-0221	Погрешность таймера должна составлять 1 мин.
279	R-0282	Цена деления шкалы таймера 2 мин	235	R-0238	Погрешность таймера должна составлять 0.5 мин.

Fig. 7. The result of the work of k-means

R-0238. Let's start by checking the k-means clustering, which is visualized in the fig. 6 (see the 3rd side of cover).

We will evaluate the result of clustering (both this and the others) as follows: all conflicting requirements must be in one cluster, other requirements either should not exist at all, or their number must be minimal. In the case of k-means, we get clusters (fig. 7).

In such a degenerate case, the algorithm was able to identify in one cluster all the contradictions of group 1 (3/3) and nothing more. No other requirements that do not tell about the division price of the timer scale were not found in this cluster. However, for the second group, the circumstances were not so successful. We managed to identify two (2/3) contradictions, but instead of the third, a "normal" requirement was added to the group. Thus, the algorithm made a mistake once.

Now consider the EM algorithm, the clustering of which is presented at fig. 8 (see the 3rd side of cover).

The EM algorithm specializes in separating mixtures. Indeed, if we take a close look at the two images, comparing them with k-means, we can see how EM is trying to divide some volumetric clusters into several (perhaps even when it is not needed). For example, the cluster of points at the coordinate (−20, 20) was determined by k-means in one cluster, while EM tried to restore the distribution over mixed clusters. When trying to identify inconsistencies, the EM-algorithm correctly identifies group 1 (3/3, 0 false), however, compared to the previous one, it adds two extra requirements to the second cluster, determining only two true (2/3, 3 false) (fig. 9).

Probably, the test specification of the requirements does not contain a mixture of clusters, and therefore the legitimacy of the use of clustering, which specifies the separation of mixtures, is called into question.

The next clustering model is agglomerative hierarchical clustering, which aims to separate nested clusters. The first option is that the distance between clusters is the average distance between all elements.

The fig. 10 (see the 3rd side of cover) shows just such a variant of hierarchical clustering. As for the results, then (3/3, 0 false) for the first group, (2/3, 4 false) for the second (fig. 11). Again, the first group can be dealt with quite easily, while the errors for the second only grow. Apparently, there are no hierarchical clusters in the specification.

If we begin to define the distance between clusters as the smallest distance between its elements, then such clustering will lead to the situation at fig. 12, see the 3rd side of cover. Already from the rendered picture, it becomes clear that no positive results can be expected from such a setting — the mechanism for determining the boundaries of clusters, based on the position that the distance between the clusters coincides with the distance of the nearest elements, leads to the fact that the clusters collapse very quickly, and that's it. the timer requirements fell into one large cluster of 80 objects. There is no sense in such clustering.

	ID	Requirement		ID	Requirement
221	R-0224	Цена деления таймера должна составлять 1 минуту	209	R-0212	Рабочий угол таймера должен составлять 360 гра...
275	R-0278	Цена деления шкалы таймера 1 мин	218	R-0221	Погрешность таймера должна составлять 1 мин.
279	R-0282	Цена деления шкалы таймера 2 мин	229	R-0232	Коэффициент оптического контраста циферблата п...
			235	R-0238	Погрешность таймера должна составлять 0.5 мин.
			276	R-0279	Контрастность шкалы к корпусу таймера должна с...

Fig. 9. The result of the EM algorithm

	ID	Requirement		ID	Requirement
221	R-0224	Цена деления таймера должна составлять 1 минуту	209	R-0212	Рабочий угол таймера должен составлять 360 гра...
275	R-0278	Цена деления шкалы таймера 1 мин	218	R-0221	Погрешность таймера должна составлять 1 мин.
279	R-0282	Цена деления шкалы таймера 2 мин	229	R-0232	Коэффициент оптического контраста циферблата п...
			235	R-0238	Погрешность таймера должна составлять 0.5 мин.
			236	R-0239	Корпус таймера должен иметь гидрофобное покрытие
			276	R-0279	Контрастность шкалы к корпусу таймера должна с...

Fig.11. The result of the Agglomerative Average

If we go from the opposite, to maximize the distance between the clusters, then the algorithm will try to "pull apart" the clusters as far from each other as possible (fig. 13, see the 3rd side of cover).

And already these results may be of interest to us. The algorithm made a mistake once in the search for group (3/3, 1 false), and three times in the second group (2/3, 4 false). As you can see (fig. 14), this setting of agglomerative clustering is an order of magnitude better, however, it is far from perfect, which raises a natural question — is there any point in implementing it.

The latest implementation of this algorithm is Agglomerative Ward. The model tries to minimize the number of combined clusters, and in theory, the picture observed with this algorithm should be the opposite of Agglomerative Single (fig. 15, see the 4th side of cover).

The result of this algorithm is shown in the fig. 16. The totals table looks pretty good: (3/3, 1 false), (2/3, 1 false). In general, this result is comparable to the k-means algorithm.

If we compare all the implementations of agglomerative clustering presented above, we can say that plus or minus the results are similar (if we do not take into account, of course, the algorithm in which the distance was measured as the minimum distance between cluster objects). However, one cannot achieve one hundred percent result, perhaps nested clusters are also not a picture that corresponds to the test specification.

And finally, the DBSCAN algorithm was considered (fig. 17, see the 4th side of cover).

Frankly speaking, it was clear in advance that this algorithm would not give the desired result — since it is based on the density of a distributed quantity, and in our case it cannot be argued that the density will be constant anywhere. Therefore, it turned out that again a large number of requirements were in one cluster, and such clustering is not informative. Yes, the user will learn the obvious — all requirements for a timer belong to one semantic group. However, finding contradictions among the 83 requirements is a very non-trivial task, and it cannot be said that in this case the goal of the system will be achieved, because the user will not significantly reduce the time for checking the requirements specification.

An intermediate result can be summed up: after testing all clustering options on the TF-IDF model, it turned out that the k-means method would be the best option. However, this method did not quite correctly identify group 2, so we will try to replace the model for transforming sentences into vectors, although it is worth noting here that TF-IDF has shown itself very well at the level of trivial examples.

The first option is the simple cosine distance method (fig. 18, see the 4th side of cover).

This method is a lightweight version of the TF-IDF algorithm. It does not take into account repetitions, it does not take into account "garbage" words, i. e. those words that occur in most sentences, and of course, it cannot be taken seriously. The picture in the figure above does not imply clustering at all — the points are almost evenly distributed on the surface, and no clusters are visually observed. Even on the basis of visual observation, it can be concluded that this model is unsuitable — and the results confirm this assumption. 43 requirements end up in the cluster, where group 1 was lucky enough to get, and 22 requirements, respectively, in group 2. This is one of those models that cannot be used in any form and was presented here purely out of its simplicity to facilitate understanding of the process.

The next model is doc2vec (fig. 19, see the 4th side of cover). And here we are faced with the first surprise — doc2vec does not show results. Regardless of whether to lemmatize sentences or not, whether to expose a distributed memory method or bags of words, the results leave much to be desired. Here is one option — DBOW (fig. 20).

Yes, both clusters speak about one thing — about the timer, but in the first case we have not a single revealed contradiction (1/3, 14 false) and a huge number of requirements that do not contradict the given one. In the second case, the picture is somewhat more pleasant — two

	ID	Requirement		ID	Requirement
220	R-0223	По диаметру таймера должны быть расположны де...	155	R-0158	Соотношение ширины полос в соседних группах до...
221	R-0224	Цена деления таймера должна составлять 1 минуту	209	R-0212	Рабочий угол таймера должен составлять 360 гра...
275	R-0278	Цена деления шкалы таймера 1 мин	218	R-0221	Погрешность таймера должна составлять 1 мин.
279	R-0282	Цена деления шкалы таймера 2 мин	229	R-0232	Коэффициент оптического контраста циферблата п...
			235	R-0238	Погрешность таймера должна составлять 0.5 мин.

Fig. 14. Result of Agglomerative Complete

	ID	Requirement		ID	Requirement
220	R-0223	По диаметру таймера должны быть расположны де...	209	R-0212	Рабочий угол таймера должен составлять 360 гра...
221	R-0224	Цена деления таймера должна составлять 1 минуту	218	R-0221	Погрешность таймера должна составлять 1 мин.
275	R-0278	Цена деления шкалы таймера 1 мин	235	R-0238	Погрешность таймера должна составлять 0.5 мин.
279	R-0282	Цена деления шкалы таймера 2 мин			

Fig. 16. The result of the Agglomerative Ward

	ID	Requirement		ID	Requirement
21	R-0022	СрЗИ не должны ухудшать функциональные характе...	210	R-0213	У таймера должна быть возможность дозаводки
217	R-0220	Срок эксплуатации таймера должен составлять не...	213	R-0216	Нижняя часть таймера должна быть аппаратной
220	R-0223	По диаметру таймера должны быть расположны де...	214	R-0217	Верхняя часть таймера должна быть подвижной
223	R-0226	На циферблате должны быть использованы арабски...	218	R-0221	Погрешность таймера должна составлять 1 мин.
224	R-0227	Циферблат должен располагаться на нижней части...	231	R-0234	Таймер должен быть цилиндрической формы
230	R-0233	Цвет стрелки должен соответствовать цвету цифе...	235	R-0238	Погрешность таймера должна составлять 0.5 мин.
232	R-0235	Диаметр таймера не должен превышать 6 см	238	R-0241	У таймера должна быть нижняя часть
233	R-0236	Высота таймера не должна превышать 7 см	269	R-0272	Таймер должен заводиться поворотом ручки
236	R-0239	Корпус таймера должен иметь гидрофобное покрытие	270	R-0273	Таймер должен отсчитывать время
247	R-0250	У таймера должна быть защита механизма от внеш...			
263	R-0266	Звук отсчета времени должен быть слышен (посек...			
266	R-0269	Габариты таймера не должны превышать 10x10x10 ...			
273	R-0276	Таймер должен иметь возможность досрочного отк...			
279	R-0282	Цена деления шкалы таймера 2 мин			

Fig. 20. Doc2Vec DBOW Results

	ID	Requirement
206	R-0209	Длительность звукового сигнала должна составля...
218	R-0221	Погрешность таймера должна составлять 1 мин.
221	R-0224	Цена деления таймера должна составлять 1 минуту
235	R-0238	Погрешность таймера должна составлять 0.5 мин.
274	R-0277	Таймер должен производить отсчет времени с пог...
275	R-0278	Цена деления шкалы таймера 1 мин
279	R-0282	Цена деления шкалы таймера 2 мин

Fig. 22. Result of BERT

contradictions have been found, but the errors are still significant (2/3, 7 false).

What is the reason for this? The situation is, in fact, extremely simple — we are using the untrained doc2vec model. First, it needs to be trained on our requirements, and 280 pieces are negligible, even if you use cross-validation. This is why the model lacks the information to build a vocabulary, and therefore the results come out so unimpressive. But here it is impossible to operate only with the fact that the specification is not voluminous enough. There are also small projects, and in such cases the number of requirements is estimated at several hundred, which means the algorithm will not work. We need to take an already trained model for our calculations.

And there is such a model — we will use BERT trained on Wikipedia data, etc. It is enough to load the already trained neural network (moreover, it is configured just to search for duplicates), and you can start getting vectors for sentences from it (fig. 21, see the 4th side of cover).

Already from the illustration there is a feeling that the results will be successful. In the picture above, clusters are visible, the points do not overlap each other strongly enough, which is a good signal. The search for clusters brings amazing results: both groups end up in the same cluster and in full. This algorithm was able to determine both simple contradictions associated with a change in one digit in the text of the requirement, and reformulation. This is the first algorithm that determined (6/6) with only one error — BERT added the R-0209 requirement to the cluster, which does not contradict anything (fig. 22).

Conclusion

In the process of writing the article, an analysis was made of clustering methods and methods for converting sentences into vectors.

The analysis was carried out as follows: contradictions were deliberately introduced into the requirements specification, after which all possible algorithms for converting sentences into vectors with clustering models were launched. As a result, the most successful options were identified that are suitable for use in industrial requirements development. The table 2 illustrates the possibility of using a combination of the two models. The

NLP-clustering intersections that are not recommended for work are marked in black, the combinations that have demonstrated high results are in striped.

Table 2

Applicability of models for verification of requirements

Vectorizing Algorithm	Clustering Algorithm						
	k-means	em	aggloaver	agglomin	agglomax	aggloward	dbscan
cosine							
TF-TDF							
doc2vecdbow							
doc2vecdm							
BERT							

In the future, it is planned to continue working on the topic. The implementation of DeepPavlov [17], Fasttext [18] and the search for other models for converting sentences into numeric vectors are already planned. Already from this work, it became clear that models based not on the frequency of the presence of words in a sentence, but on deeply trained neural networks, will show themselves best. Therefore, an additional analysis of the available options is required. It is also possible to find a pre-trained doc2vec model, or to conduct an analysis on a specification consisting of several thousand requirements and achieve a positive result.

With regard to the interface, it is planned to develop a plugin for Google Docs. It often happens that the requirements are stored in the cloud as a table in a .docx file. Google Docs' Check Requirement button would make life easier for analysts, if only because it would not require them to upload requirements specification into.csv format.

References

1. **Hall E.** *Requirements Engineering*, US, Kent, Gray Publishing, 2005, 239 p.

2. **Batovrin V., Gaydamaka K.** Requirements engineering — key factor for project success, *The Project Management Journal*, 2017, no. 1 (49), pp. 6—20.

3. **Chatzipetrou P., Unterkalmsteiner M., Gorschek T.** Requirements' Characteristics: How do they Impact on Project Budget in a Systems Engineering Context? *In2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* 2019 Aug 28, IEEE, 2019, pp. 260—267.

4. **INCOSE,** Guide for writing requirements INCOSE TP-2010-006-02, USA, San Diego, International Council on Systems Engineering, 2015, 73 p.

5. **Jordan M. I., Mitchell T. M.** Machine learning: Trends, perspectives, and prospects, *Science*, 2015, vol. 349, no. 6245, pp. 255—260.

6. **Muhamedyev R.** Machine learning methods: An overview, *Computer modelling & new technologies*, 2015, vol. 19, no. 6, pp. 14—29.

7. **Hartigan J. A., Wong M. A.** Algorithm AS 136: A k-means clustering algorithm, *Journal of the royal statistical society. Series C (applied statistics),* 1979, vol. 28, no. 1, pp. 100—108.

8. **Moon T. K.** The expectation-maximization algorithm, *IEEE Signal processing magazine,* 1996, vol. 13, no. 6, pp. 47—60.

9. **Zhao Y., Karypis G., Fayyad U.** Hierarchical clustering algorithms for document datasets, *Data mining and knowledge discovery*, 2005, vol. 10, no. 2, pp. 141—68.

10. **Schubert E., Sander J., Ester M., Kriegel H. P., Xu X.** DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems,* 2017, vol. 42, no. 3, article 19, pp. 1—21.

11. **Sravanthi P., Srinivasu B.** Semantic similarity between sentences, *International Research Journal of Engineering and Technology (IRJET),* 2017, vol. 4, no. 1, pp. 156—61.

12. **Zhang Y., Jin R., Zhou Z. H.** Understanding bag-of-words model: a statistical framework, *International Journal of Machine Learning and Cybernetics*, 2010, vol. 1, pp. 43—52.

13. **Ramos J.** Using TF-IDF to determine word relevance in document queries, *Proceedings of the first instructional conference on machine learning 2003*, 2003, 4 p.

14. **Rong X.** word2vec parameter learning explained. arXiv preprint arXiv:1411.2738. 2014 Nov 11.

15. **Lau J. H., Baldwin T.** An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:1607.05368. 2016 Jul 19.

16. **Devlin J., Chang M. W., Lee K., Toutanova K.** Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 2018 Oct 11.

17. **Burtsev M., Seliverstov A., Airapetyan R., Arkhipov M., Baymurzina D., Bushkov N., Gureenkova O., Khakhulin T., Kuratov Y., Kuznetsov D., Litinsky A.** DeepPavlov: Open-source library for dialogue systems, *Proceedings of ACL 2018*, System Demonstrations 2018 Jul., 2018, pp. 122—127.

18. **Joulin A., Grave E., Bojanowski P., Douze M., Jégou H., Mikolov T.** Fasttext. zip: Compressing text classification models. 2016. arXiv preprint arXiv:1612.03651.

Рисунки к статье

K. I. Gaydamaka, A. D. Belonogova

«APPLYING UNSUPERVISED MACHINE LEARNING ALGORITHMS TO ENSURE REQUIREMENTS CONSISTENCY»

Fig. 2. K-means clustering

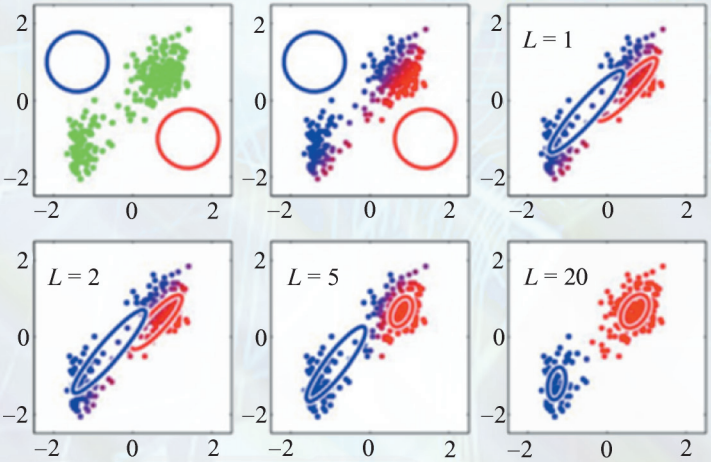

Fig. 3. EM Clustering

Рисунки к статье **K. I. Gaydamaka, A. D. Belonogova**
«APPLYING UNSUPERVISED MACHINE LEARNING ALGORITHMS TO ENSURE REQUIREMENTS CONSISTENCY»

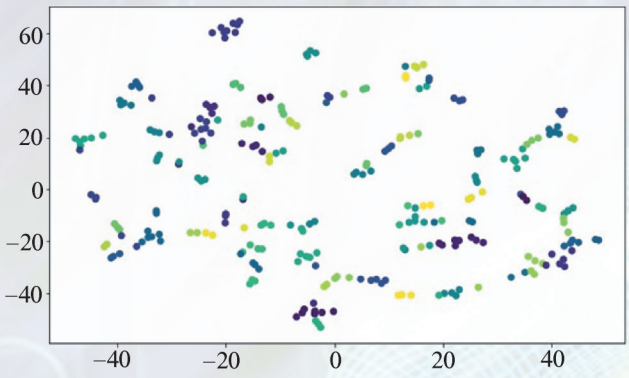

Fig. 6. K-means clustering

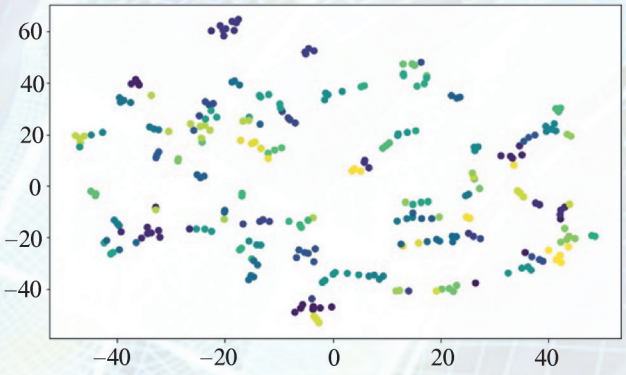

Fig. 8. EM clustering

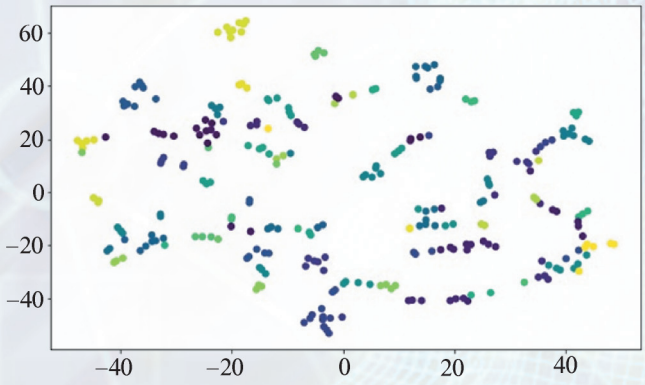

Fig. 10. Clustering Agglomerative Average

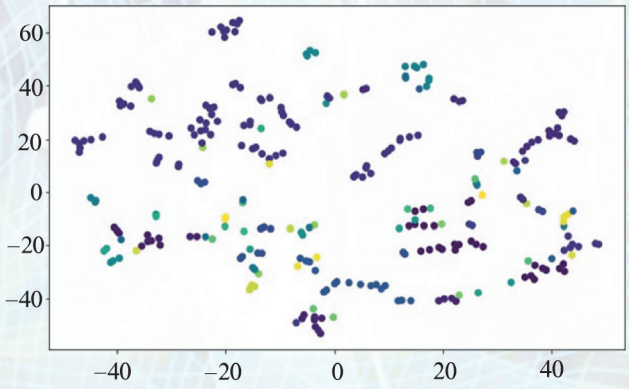

Fig. 12. Clustering Agglomerative Single

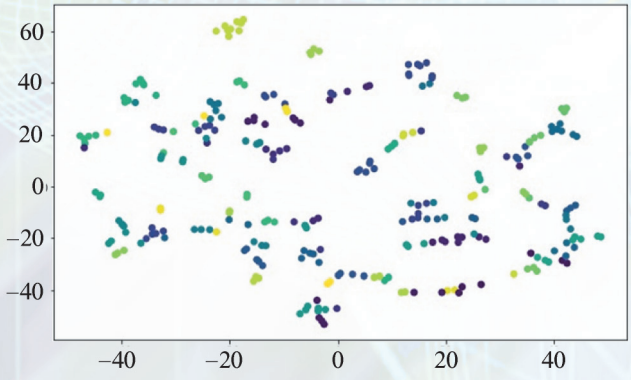

Fig.13. Clustering Agglomerative Complete

Рисунки к статье **K. I. Gaydamaka, A. D. Belonogova**
«APPLYING UNSUPERVISED MACHINE LEARNING ALGORITHMS TO ENSURE REQUIREMENTS CONSISTENCY»

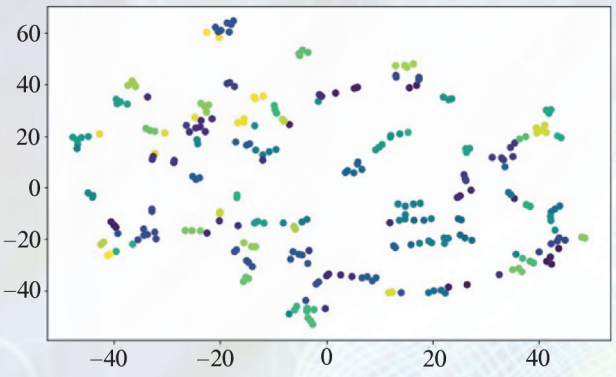

Fig. 15. Clustering Agglomerative Ward

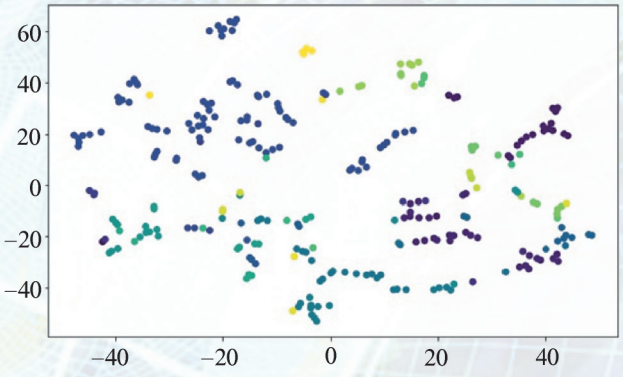

Fig. 17. DBSCAN clustering

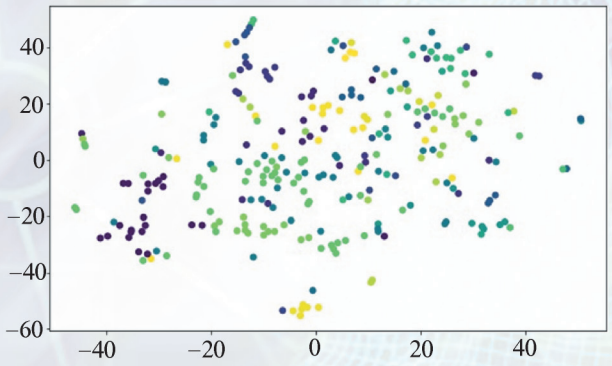

Fig. 18. Clustering the method of simple cosine distances

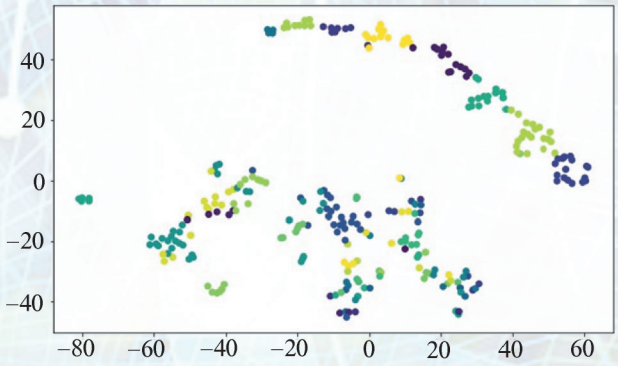

Fig. 19. Doc2Vec DBOW clustering

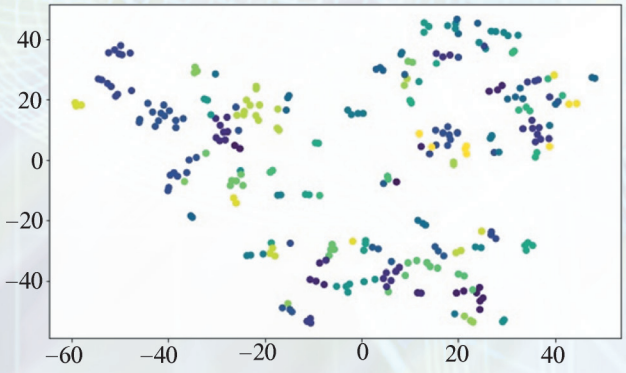

Fig. 21. BERT clustering

ISSN 2220-3397. Программная инженерия. 2022. Том 13. № 4. 153 – 208. Подписной индекс по Объединенному каталогу "Пресса России" – 22765