A Method for Facilitating Inventive Design Based on Semantic Similarity and Case-Based Reasoning

Wei Yan\textsuperscript{a}\textsuperscript{*}, Cecilia Zanni-Merk\textsuperscript{b}, François Rousselot\textsuperscript{b}, Denis Cavallucci\textsuperscript{a}

\textsuperscript{a} LGECO / INSA Strasbourg, 24 Boulevard de la Victoire, 67084 Strasbourg Cedex, France
\textsuperscript{b} LSIIT / FDBT Team (UMR CNRS 7005) – Pôle API BP 10413, 67412 Illkirch Cedex, France

Abstract

While TRIZ is increasingly developing both in research and education, new users always encounter difficulties in their first attempts to practice it. In such situation, Altshuller’s original matrix often appears as an “easy-to-begin-with” tool. However, while not being representative of what TRIZ really is, it continues to seduce new users, teachers and trainers. Several approaches to automate the use of the contradiction matrix have been proposed in literature, and researches on the automatic match between specific parameters of the artefact being considered and Altshuller’s generalized parameters remains a topic of interest. In this paper, we propose a way to automate the proposal of equivalencies between a specific parameter and a generic parameter from the matrix. This method calculates the semantic distance between short texts, and uses it to fill the semantic gap between the specific parameters of the artefact and the generalized ones. Case-based reasoning is also used to improve the whole accuracy of the method.

Keywords: Ontology; Semantic distance; Information content; WordNet; Case-based reasoning.

1. Introduction

The inventive design methodology we are interested in, TRIZ (Theory of Inventive Problem Solving) [1] [2], is primarily about technical and physical problems, but is now being used on almost any problem or situation. The key to success in TRIZ is the fact that (technical) systems evolve in similar ways, and, by reducing any situation and problem to an abstract level independent of any domain, almost standard solutions and problem solving techniques can be applied, even coming from other industries.

The first TRIZ problem solving technique was a collection of Inventive Principles aimed at eliminating technical contradictions. To make the inventive principles applicable in a systematic way, the creator of TRIZ, G. Altshuller [3] formulated 39 generalized parameters, such as "the weight of a moving object" or "speed". These are generalizations of specific technical parameters. TRIZ provides a tool called
Contradiction Matrix, which presents all contradiction parameters in the form of a matrix. It consists in 39 generalized parameters on the first line and column of it. Each cell of the matrix shows the inventive principles that can be used to solve that particular contradiction. Along the vertical axis of this matrix, the generalized parameters that have to be improved are specified. Along the horizontal axis, the parameters that deteriorate because of the improvement are specified. These parameters can be looked up along the vertical and horizontal axes and the matrix suggests up to four sorted principles that can be used to solve the contradiction.

In real-world problems, most of the times, contradictions are established in terms of parameters that are inherent to the artefact that is being developed, and there is a semantic gap to fill between those parameters and the generalized ones. An abstraction effort needs to be provided to choose the best generalized parameter, and in this way, be able to use the contradiction matrix.

In the framework of an inventive design project proposed to a class of engineering students in our school, there was the study of the improvement of a barbecue grill. The students have decided to solve the following contradiction: if the number of parts in the wire mesh is high, the mastery of the beef doneness is satisfying but the weight of the grill is unsatisfying. On the other hand, if the number of parts is low, the mastery of the beef doneness is unsatisfying but the weight is satisfying. We have then two parameters: PE1, the weight; PE2, the mastery of the doneness.

PE1 is directly associated with the second generalized parameter "weight of a stationary object"; but for PE2, the association with the 35th generalized parameter "adaptability or versatility" is not intuitive.

As we have seen, the generalized parameters in the Classical Matrix are abstract and built independently of specific applications. In real applications, it is difficult for users to match specific parameters to the 39 generalized parameters because it requires an extensive knowledge of different engineering domains.

We make the hypothesis that semantic technologies may be used to fill the gap between real-world problems and the high-level abstract concepts manipulated by TRIZ. In order to fill the gap between the real parameters and the generalized ones and to facilitate the process of using the contradiction matrix, we therefore present here, in section 2, a literature review about different ways to cope with this problem, which proves the necessity of our research. In section 3, we present our proposal, and in particular, after a short state of the art on the measurement of semantic distance and case-based reasoning, we present our method in detail. Finally, section 4 presents some conclusions and perspectives of future work.

2. Literature review

With the development of TRIZ, several authors have intended to automate the use of the contradiction matrix with different approaches. For example, Te and Hsing [4] proposed to employ a fuzzy analytical hierarchy process as a decision support tool to automate inventive design, and Von WS et al [5] explored a cooperative multi-agent platform to support the invention process based on the patent document analysis and ontologies. In different directions, but nevertheless complementary to ours, Duflou et al [6] have worked on the contradiction identification in inventive design, while Coelho proposed to match engineering parameters to human factors in industry [7].

Altshuller’s Classical Matrix was developed more than 50 years ago using patents from traditional engineering systems. Subsequent studies indicated that it was difficult to use the Classical Matrix to solve problems in different fields. For example, Sheu et al [8] established a set of engineering parameters, innovative inventive principles and Classical Matrix prototypes for chemical mechanical polishing because of that.

In order to use the inventive principles to solve problems in different fields, several researches were done in different directions.
He et al [9] grouped 40 inventive principles according to text similarity and meaning similarity, and used them to facilitate the automatic classification of patent documents. Keeping the Classical Matrix unchanged, for all previous patent documents, they obtained automatically the corresponding classification results and for each new problem, they easily sorted it and reused the solutions of patents of the same kind. Compared with this method, our approach is also carried out based on the Classical Matrix. However, in contrast to the automatic classification of patent documents, we aim to explore a method to facilitate the process of solving problems based on inventive principles, in which the matching between the specific parameters and 39 generalized parameters is the most critical task. As a result, we design a method to calculate their semantic similarity and use the technique of case-based reasoning to improve its performance.

Sheu et al [10] proposed a TRIZ problem-solving model for multiple-to-multiple parameter contradiction using case-based reasoning. They developed a contradiction matrix in the semiconductor industry, especially in the chemical-mechanical polishing domain. The engineering parameters and the improved inventive principles, explored by Sheu [8], were used to construct a new contradiction matrix based on a large number of patent cases. For each new case, they firstly calculated the similarity between the engineering parameters of the new case and that of previous case. Then, these similarities were used to compute the weight of each inventive principle the new case used. Two different results can be obtained from this approach:

1. The most appropriate inventive principle to use for the new case, based on similarity measures;
2. The inventive principles used to solve the most similar case in the database.

In our method, case-based reasoning is also used. However, our approach is very different from Sheu’s. We prefer to automate the process of solving problems based on the Classical Matrix rather than reconstruct a new contradiction matrix, that is, to explore a method to calculate the semantic similarity between the specific parameters and the 39 generalized ones and to use the technique of case-based reasoning to improve its performance.

3. Our proposal

We explore a method to calculate the semantic distance between short texts (sentences in the range of 10-20 words using correct grammar) and use it to fill the semantic gap between the specific parameters and the generalized parameters, and in this way, facilitate the process of using the contradiction matrix. Then we reuse the experience in previous projects to improve whole accuracy, by using case-based reasoning technology.

3.1. The method to calculate semantic similarity

Semantic similarity or semantic relatedness is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning / semantic content.

The literature presents several surveys on measures of semantic relatedness, in particular, [11] presents an extensive state of the art and classification. We are interested in the measures that use WordNet [12], a large lexical database of English, as a knowledge base. These methods vary from simple edge counting [13] to attempts to calculate taking into account certain characteristics of the structure of WordNet by considering the link direction [12], the relative depth [14] or the density [15]. There are also other methods using statistical and machine learning techniques. Finally, there are hybrid approaches combining different knowledge sources ([16], [17] or [18]).
The development of the semantic similarity measurement promotes its wide applications in different domains. The method based on corpus statistics and lexical taxonomy is used widely in the Information Retrieval Systems and Natural Languages Processing (NLP). In the geospatial field, the semantic similarity assessment, which combines the advantages of the feature-matching process and the semantic distance calculation, is proposed to measure the degree of interoperability [19]. In the medical domain, a measurement of semantic relatedness based on context vectors derived from medical corpora is explored to facilitate the process of recognizing diseases [20].

Information Content (IC) is the amount of information represented by an item. It is an important quality when assessing the similarity of two terms or word senses. The classical method of measuring the IC of word senses is to combine knowledge of their hierarchical structure with statistics on their actual usage in text as derived from a large corpus [21]. In our proposal, the IC we use is:

\[
IC(c) = 1 - \frac{\log(hypo(c)+1)}{\log(max_{wn})}
\]  

where \(c\) represents a concept, \(hypo(c)\) returns the number of hyponym concepts of concept \(c\) in WordNet, and \(max_{wn}\) is the number of concepts in WordNet.

In the framework we are considering, we usually need to calculate the semantic similarity between short texts, such as the text description of the inventive principles or of the parameters. Therefore, we present here a specific method to calculate semantic similarity among short texts, which includes the following five steps:

- **Word Segmentation**: We need to divide the short text into several words by using techniques of word segmentation, such as Tokenization (e.g., substance appearance - disappearance → <substance, appearance, disappearance>), Lemmatization (i.e., copies → copy) and Elimination (e.g., remove ’a’, ’by’, ’my’, ’to’);
- **Sense Search**: For each word obtained, we use WordNet to look for their corresponding senses, including nouns, verbs, adjectives and adverbs. For example, the noun form of the word ’way’ has twelve different senses;
- **Sense Similarity**: We use Lin’s measure [18] to calculate the semantic similarity of the senses of two words. With this measure, it is obvious that in WordNet, the higher the rate of sharing information, the more similar the two concepts;
- **Word Similarity**: We choose the maximum sense similarity of the two words as their word similarity;
- **Short text Similarity**: Short text similarity is calculated based on word similarity. We assume that two sentences: \(A, \ A_1, A_2 \ldots \ A_m\) and \(B, \ B_1, B_2 \ldots \ B_n\) represents word similarity of \(A_i\) and \(B_j\), \(1 \leq i \leq m, 1 \leq j \leq n\). We can build the matrix \(M(A,B)\):

\[
\begin{bmatrix}
  s(A_1, B_1) & s(A_1, B_2) & \cdots & s(A_1, B_n) \\
  \vdots & \vdots & & \vdots \\
  s(A_m, B_1) & s(A_m, B_2) & \cdots & s(A_m, B_n)
\end{bmatrix}
\]

(2)

We can use this matrix to obtain the semantic similarity of two sentences \(A\) and \(B\):

\[
s(A, B) = \frac{\sum_{i=1}^{m} \max(s(A_1, B_1), \ldots, s(A_i, B_n))}{m}
\]  

(3)
To test the accuracy of the semantic similarity between two short texts that we have retained, we have worked on projects that were solved by engineering students in our school. In this framework, we have tried to match 156 specific parameters of these projects and the 39 Altshuller’s generalized parameters.

In the following, \( \text{result1} \) represents the set of matches done by the students and \( \text{result2} \), the set of results coming from the automatic semantic similarity calculation.

For example, in the project about the improving of a “pan”, the students manually matched the specific parameter “several different foods to cook” to the generalized parameters “26. Quantity of substance/the matter” and “35. Adaptability or versatility”. With our method, we automatically match this specific parameter with “23. Loss of substance” and “26. Quantity of substance/the matter”.

We have identified six different situations:

- a. \( \text{result1} = \text{result2} \): \( \text{result1} \) equals to \( \text{result2} \) exactly;
- b. \( \text{result1} \subset \text{result2} \): \( \text{result2} \) contains \( \text{result1} \);
- c. \( \text{result1} \cap \text{result2} \neq \emptyset \): There is an intersection of \( \text{result1} \) and \( \text{result2} \);
- d. \( \text{result1} \supset \text{result2} \): \( \text{result1} \) contains \( \text{result2} \);
- e. \( \text{result1} \cap \text{result2} = \emptyset \): There is no intersection of \( \text{result1} \) and \( \text{result2} \);
- f. \( \text{result2} = \emptyset \): \( \text{result2} \) is empty.

The ratio of each kind of comparison results to the total number in our experiments is shown in the following table (Table 1).

<table>
<thead>
<tr>
<th>Type of comparison results</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) ( \text{result1} = \text{result2} )</td>
<td>19%</td>
</tr>
<tr>
<td>B) ( \text{result1} \subset \text{result2} )</td>
<td>21%</td>
</tr>
<tr>
<td>C) ( \text{result1} \cap \text{result2} \neq \emptyset )</td>
<td>3%</td>
</tr>
<tr>
<td>D) ( \text{result1} \supset \text{result2} )</td>
<td>4%</td>
</tr>
<tr>
<td>E) ( \text{result1} \cap \text{result2} = \emptyset )</td>
<td>45%</td>
</tr>
<tr>
<td>F) ( \text{result2} = \emptyset )</td>
<td>8%</td>
</tr>
</tbody>
</table>

Intuitively, we would like to have the set A) as big as possible, while having sets E) and F) as small as possible. Results of sets B), C) and F) might be acceptable to some extent; but we would like to decrease their ratios to improve the ratio of set A).

The results in Table 1 are not concluding. We can see that set A) represents only 19% of the whole, while sets E) and F) represent 45% and 8% respectively. It means that for 53% projects, the results of the automatic approach do not correspond to the matches done by the students. Only for 19% projects, we get all the exact matches. For the rest, we obtain more or less correct solutions.

The analysis of this situation leads to three reasons resulting in it:

- The inaccurate description of the projects: In our experiments, we assume that all elements in \( \text{result1} \) are correct. In fact, there are differences among the views of different people about the same project; not all the students depict the project accurately. As a result, there are some obvious mistakes in the matches retained by the students. This is also the main reasons why the size of set E) is so big.

- The restriction of the dictionary: We use WordNet2.0 as our dictionary because it is one of the most comprehensive dictionaries. As an electronic dictionary, it provides an interface to facilitate the programming process based on it. However, the number of meanings of the words is limited. That is the reason for the appearance of the set of type F).
• **Problems with our method:** We explore the method to calculate the semantic similarity between short texts. This calculation is done without considering contexts. By “context”, we mean the area or field of the project. For instance, in the case of the artefact “instantaneous water heater” the specific parameter “size” is matched with “7. Volume of moving object”. In the case of the artefact “sheet music stand”, the same parameter “size” is matched with “8. Volume of stationary object”. Obviously, these two cases have different contexts. Without considering contexts, our method does not have a good performance for the specific application in inventive design.

  To cope with these disadvantages, in section 3.2 we propose an improvement of the method we have presented so far.

### 3.2. The improved version of the method: Semantic similarity + case-based reasoning

Case-based reasoning (CBR) [22] is a well-known methodology in Artificial Intelligence (AI). The main idea of CBR is to adapt solutions that were used to solve old problems and use them for solving new problems or cases, that is, people reuse past problem-solving experiences to deal with a new case.

When we use CBR to solve a new problem, firstly the new problem is described according to a specific mode. Then similar cases are retrieved from the case library, and the solution of the most similar problem is suggested for the new problem. If necessary, this suggested solution is revised according to previous experience, domain knowledge and the actual situation of the new problem. After we obtain the confirmed solution, we store the new case, composed of the new problem and its confirmed solution into the case library.

The cycle of CBR comprises five activities:

- Characterise the new problem and assign an index;
- Retrieve the most similar case(s) to the new problem from the case library;
- Reuse the case(s) to attempt to solve the new problem;
- Revise or adapt the suggested solution to satisfy the new problem if necessary;
- Retain the new problem and its confirmed solution as a new case.

The cyclic process of CBR is depicted in Fig 1 [22].

![Fig. 1. The CBR cycle](image-url)
According to the above five activities, we construct specific a case-based reasoning framework to improve our method. Specifically, on the one hand, we use our way to calculate the semantic similarity between short texts to return the matched generalized parameters. On the other hand, we use case-based reasoning to retrieve the solution to the most similar previous problem and return it to users as a suggested solution simultaneously. The whole flowchart of the improved method is described in Fig. 2.

![Flowchart of the improved method](image)

**Fig. 2. The whole flowchart of the improved method**

**Case representation:** Case representation is an important aspect in designing efficient CBR systems. In our application, we represent a case with the following three attributes:

- The context number;
- The match from the specific parameter (that is, one of the parameters of the artefact involved in the contradictions) to a generalized parameter (we choose these matches as the previous experience and reuse them because they have been used under the specific context), as shown in Fig 3.

<table>
<thead>
<tr>
<th>Case Name</th>
<th>Context Number</th>
<th>Specific Parameter</th>
<th>Generalized Parameter</th>
</tr>
</thead>
</table>

**Fig. 3. The case representation**

In order to make a comprehensive classification of contexts, we use the US patent classes classification [23]. All the contexts are divided into four groups:

- Chemical and related arts;
- Communications, radiant energy, weapons, electrical, and computer arts;
- Body treatment and care, heating and cooling material handling and treatment, mechanical manufacturing, mechanical power, static and related arts;
Industrial design.

Each group is composed of several classes, and these classes are made up of numerous subclasses, organized in a hierarchy of seven levels. For example, we have, “Class 131: Tobacco”, including all the aspects concerning tobacco, and its corresponding Subclasses, such as “Subclass 270: Antismoking product or device” and “Subclass 274: Product or device having identifiable constituent to flavour or enhance flavour”. Unfortunately, classes and subclasses are indexed in a quite chaotic way; anyway, we will use their index as the context identifier, as this association is unique.

When the user builds the representation of a case, he needs to choose a context index number for it, but sometimes, he may not find an exact match. If it is the case, the user should look for terms of approximately the same meaning, for terms of either broader or narrower scope, or for terms that represent a different approach to the subject; i.e., the essential function or effect of the device or the use or application to which the device is put.

Case library and case index: We represent all past cases according to the above framework, and then index them by the unique identifier (the context number) to facilitate the retrieval.

We permit that a specific parameter with similar context corresponds to several generalized parameters, that is, for a new problem; we may obtain several different suggested results based on case-based reasoning.

Case retrieval: It includes the retrieval of past similar cases and the selection of the best case. Firstly, we choose a context number for the new case. Then we query the case library by comparing the specific parameter of the new case and the previous case. For cases whose specific parameter is the same as the new case, we next compare their context numbers, and decide whether they have a similar context. Finally, the solutions of cases with the similar context, that is their matches between the specific parameter and the generalized parameter, are returned to users as suggested results.

The process of searching the most similar context plays an important role in our improved method. Firstly, we construct a group-class-subclass tree to represent the classification of contexts with different granularities, larger for groups and smaller for subclasses. For a given case, we choose the index of the subclass - the minimal granularity to represent it. Then we apply a bottom-up approach of the retrieval of the group-class-subclass tree, as shown in Fig 4 and search for the similar context according to the following steps:

1. Retrieve cases belonging to the same subclass, if they exist, go to step 4, else go to step 2;
2. Use DFS (depth first search) [24] to retrieve the cases belonging to the subclasses who have the same parent – classes with the given index of the subclass of the context, if they exist, go to step 4, else go to step 3;
3. Use DFS to retrieve cases belonging to the classes who have the same parent – group with the given index of the subclass of the context, if they exist, go to step 4, else return “no case with similar context”; 
4. Return the retrieved cases.

According to the above four steps, we retrieve at least the first minimal level of the group-class-subclass tree. Generally, we will finish our program as soon as we find the first similar case because the bottom-up retrieval along the group-class-subclass tree ensures that the first obtained case is the most similar case.

Let us consider a new case “NC” in Fig. 4. If the user chooses “Subclass 3” as the context, step 1 of the algorithm just returns “Case 3” and “Case 4” as the most similar cases. On the contrary, if the user

---

a Although the hierarchy in [23] has numerous levels, we have chosen here, for the sake of simplicity, to represent only three of them.
chooses “Subclass 4” as the context, steps 1 and 2 do not find any similar cases; our bottom-up search does not return any similar case. Another situation would be if the user chooses “Subclass 7” as the context. In this situation, we need to generalize to the upper levels (“Class 6” and “Group 4”) to retrieve “Case 6” as the most similar one to the NC. We take the matching of specific parameter “size” in the new case “instantaneous water heater” as an example. Supposed that the context of this case is “Subclass 7”, and according to the stated above, we can obtain a similar case “Case 6” with the matching between “size” and “volume of stationary object” in “Case 5” can not be returned because it has different parent even in the group-level.

Case adaptation and reuse: This step is to analyze items requesting adaptation and implement adaptation. For our application, we need the TRIZ user to choose the best match from the automatic matching results and all the suggested results.

Case verification: This part is to retain the new case. As the new problem has a new solution, it can be described according to the case representation and added into the case library. In addition, with the rapid increase of the number of cases, the case with repeated matches under a similar context should be deleted because it may result in redundant cases. The purpose is to make sure that the size of the case library would not increase continuously to affect the retrieval speed and make each problem more correct with higher accuracy.

Maintenance of the case library: Each case keeps track, in fact, of a “translation” such as specific parameter -> generalized parameter, valid in a certain context. We would like to have this context as general as possible. In other words, the idea is to find the more general context where the two short texts (that represent the specific and generalized parameters) have the same sense. This is why we to launch periodically this maintenance task of the case library. If all the leaves of the sub-tree of contexts have cases with the same translation associated to them, we can assume that those translations are valid in the more general common context. For example, let us consider the following leaves in the taxonomy of Fig. 4; “Subclass 1”, “Subclass 2”, “Subclass 3”. These three contexts have cases associated with them. Let us
suppose that “Case 1”, “Case 2” and “Case 3” have the same translation “SP1 -> GP1”. In this case, we can propose the user to validate these three translations in the context “Group 1” (the more general common context for the leaves we have considered).

4. Conclusion

In this paper, firstly we review several methods to solve the automatic use of Altshuller’s matrix that were proposed in the literature. We propose, then, a method to calculate the semantic distance between short texts and use it to fill the semantic gap that exists while trying to match a specific parameter of the artefact to develop and the generalized one. In order to improve the performance of our method, we return the results from previous projects to users as suggested solutions according to a case-based reasoning approach. As we can observe from our first attempt to test the developed methodology, some relevant potential associations between a specific term and one or several of the 39 engineering parameters are proposed to users. Our method also benefits from previous experiences and therefore offers a constant improvement due to CBR techniques.

However, two aspects have to be discussed. First, an educator would criticize our proposal by saying that easing the job of an engineer by automating part of his thinking process may result in a diminution of his skills. It is partly true, this is why we would not propose the systematic use of our method if the goals were to increase engineering knowledge or improve analogy-reasoning skills of a population of students. Nevertheless, even in such circumstances, our automated procedure can be a “backup solution” to a lack of understanding of a given situation. Second, in all other cases, that is, industrial situations or real-life situations, the method is a potential help to a better use of Altshuller’s matrix and a step forward an automated process of solving inventive problem. Future research in this area will be to test and compare at a wider scale the human-made “translations” and the results obtained with our method. In this way, we could also measure the relative importance of them with regards of the pertinence of the solutions obtained.

Another interesting possibility we are exploring is the fact that the inventive principles have links with other TRIZ knowledge bases (such as the inventive standards). We are planning to use semantic technologies to link all these knowledge bases and, in this way, take advantage of the easy use of Altshuller’s matrix to benefit also from the inventive standards to get the solution to the problem being considered.

References


