applied sciences

MDPI

Article

Automated Transformation from Competency List to Tree: Way to Competency-Based Adaptive Knowledge E-Evaluation

Asta Margienė [1], Simona Ramanauskaitė [1,*], Justas Nugaras [2] and Pavel Stefanovič [3]

[1] Department of Information Technologies, Vilnius Gediminas Technical University, LT-10223 Vilnius, Lithuania; asta.margiene@vilniustech.lt
[2] Department of Creative Communication, Vilnius Gediminas Technical University, LT-10223 Vilnius, Lithuania; justas.nugaras@vilniustech.lt
[3] Department of Information Systems, Vilnius Gediminas Technical University, LT-10223 Vilnius, Lithuania; pavel.stefanovic@vilniustech.lt
* Correspondence: simona.ramanauskaite@vilniustech.lt

Featured Application: The proposed model for competency list transformation to competency tree can be applied in e-learning and e-evaluation systems to ensure integration between systems, using competency lists and competency trees and, at the same time, providing some additional adaptability for the existing e-learning systems.

Abstract: E-learning is rapidly gaining its application. While actively adapting student-oriented learning with the competency evaluation model, the standard of competency support in existing e-learning systems is not implemented and varies. This complicated integration of different e-learning systems or transfer from one system to another might be challenging if the student had his or her competency portfolio in list form, while another system supports tree-based competency portfolios. Therefore, in this paper, we propose a transformation model dedicated to converting the competency list to a competency tree. This solution incorporates text processing and analysis, competency ranking based on Bloom's taxonomy, and competency topic area clustering. The case analysis illustrates the model's capability to generate a qualitative tree from the competency list, where the average accuracy of competency assignment to appropriate parent competency is 72%, but, in some cases, it reaches just 50%.

Keywords: competency list; competency tree; transformation; e-learning; NLP; Bloom's taxonomy; clustering

Citation: Margienė, A.; Ramanauskaitė, S.; Nugaras, J.; Stefanovič, P. Automated Transformation from Competency List to Tree: Way to Competency-Based Adaptive Knowledge E-Evaluation. *Appl. Sci.* **2022**, *12*, 1582. https://doi.org/10.3390/app12031582

Academic Editors: Juan A. Gómez-Pulido, Young Park, Ricardo Soto and José M. Lanza-Gutiérrez

Received: 8 January 2022
Accepted: 29 January 2022
Published: 1 February 2022

1. Introduction

The knowledge testing process is evolving, and student-oriented teaching, based on the estimation of the student competency portfolio, is gaining its application in schools. At the same time, e-learning and e-evaluation systems for distance or blended learning must incorporate the competencies based principles, too. However, competence management and evaluation have not yet been standardized. Competency match methodologies between different e-learning and e-evaluation systems are not designed, and competency match between different courses might be difficult to ensure.

One of the problems is the different presentation of competencies in e-learning systems. Some e-learning systems are based on a competency tree, while others use a competency list. While competency tree transformation to competency list requires no effort (nodes or just leaves of the tree are listed), the transformation from competency list to the tree is mostly done by a human as it requires context understanding and additional analysis, such as text interpretation. This slows down the integration of different e-learning systems. Therefore, this research aims to automate the competency list transformation to a competency tree for easier integration of different architecture e-learning systems.

At the same time, a transformation from a list of competencies to a tree of competencies brings an additional value both to the students and teacher—the competencies are more structured and allow an easier understanding of the knowledge and skills, how it is related, and how it is composed. By using the competency tree, the teacher can see what the sequence of teaching competency achievement should be, while a student can understand the base, necessary competencies for further, sequential knowledge and skill improvement.

In this paper, we investigate the problem of transforming the text-written competency list to the competency tree structure. This is done by applying different natural language processing (NLP) methods. The review of related works is presented in Section 2; Section 3 presents the designed method for competency list transformation to tree, while Section 4 is dedicated for the method validation, by applying the method to different situations and estimating its efficiency. The paper is summarized with research conclusions.

2. Related Works

E-learning is growing in its application. It was especially noticeable within the COVID-19 pandemic, when the learning process was moved to distance, rather than contact, form [1]. However, given the benefits of e-learning and e-evaluation solutions, it is still not capable of replacing human tutors in the sense of adaptability. In a testing process, a human tutor automatically reacts to the students' responses and makes the knowledge evaluation process adaptable, focusing on the identification of the weakest and strongest areas of the student's knowledge. Meanwhile, e-evaluation systems are still on the way to being more student knowledge-oriented and adaptive.

In the e-evaluation system, based on the assessment of the student competency portfolio, different data structures are used to define the competencies and links between them. Currently, in adaptive e-evaluation systems, three main types of data structures are used for competency presentation [2]: linear, tree-based, and graph-based. Those data structures, for competency storage, might vary in usage purposes:

- List of competencies (course objectives) and mapping to e-evaluation tasks for the implementation of personalized adaptive learning content for each student [3]. Based on the results of the e-evaluation, the competencies proven are estimated; therefore, the additional competencies required can be estimated, at the same time providing the e-learning resources for further study. However, the list of competencies does not define the sequence of the material, just a list of objectives that must be achieved.
- Tasks are categorized according to the level of difficulty into easy, medium, difficult, etc. [4–6]. In such cases, a threshold level is used, and, if a student passes it, he or she gets to a more difficult level of tasks or material. This method does not show exactly what the student does not know but is suitable for sequentially increasing difficulty e-learning.
- Aukstakalnis et al. [7] suggest that the teacher construct context graphs in adaptive knowledge assessment systems. The use of a contextual graph makes it possible to select a further task depending on whether the previous task was performed correctly or incorrectly. However, creating a context graph requires specific guides and is complicated for beginners, and a contextual graph can be difficult to follow when performing tasks that involve many contextual elements arranged on a graph [8].
- Fonseca and Faria [9] developed an adaptive assessment tool ACM-LBMCGF, using advanced concept maps, combined with the multi-choice Google Forms (MCGF) logical branching feature. Silva et al. [10] suggested, for learning assessment, to use concept maps and ontology alignment. The concept maps allow easier understanding of relations between different concepts, as well as competencies.
- A hierarchy-based competency structure, in which a competency tree is designed, can be used for adaptive and personalized e-evaluation [11]. The competence tree design method is based on context modeling of the subject content.

The variety of data structures for competency storage in e-learning systems demonstrates that there is no one superior solution for all cases. The study carried out in Ref-

erence [2] examined which of these data structures best meets the needs of students for knowledge assessment and self-assessment. Criteria for test structure, adaptability, and feedback were examined. The results showed that none of the structures fully meet the needs of students, so all three structures need to be used to create a suitable adaptive e-evaluation system.

However, Ausubel's theory of meaningful learning states that, for meaningful learning to take place, learners must master new concepts with prior knowledge by integrating them into a systemic structure [12]. Therefore, some links between competencies are preferred [13]. A hierarchical structure can be used to categorize knowledge by complexity [14,15]. Liang et al. [16] claim that a hierarchy-based structure is well-suited for assessing students with different levels of preparation. Therefore, the hierarchical, tree structure competencies have additional value in comparison to the plane list of competencies as support adaptive learning/evaluation path implementation possibilities.

For bigger integration of different e-learning systems, as well as the implementation of higher-level or e-learning adaptability and personalization, the transformation between different data structures are preferable. These types of transformation already exist—an automatic method of transformation from competency-based tree structure assessment data to a graph structure was proposed in 2021 [17]. In this case, competencies by hierarchy are stored in a competency tree-based structure, and a graph-based structure is used for adaptive task presentation. The advantage of automatic transformation from the competence tree to the contextual graph is that it is possible to create one competence tree for the whole study subject and generate different contextual graphs from it as needed. Thus, there is no need to create a competence tree nor graph manually for each case. A graph can be generated from the entire competency tree, or only from a selected part of it.

Achieving integrity between different formats of competency presentation automated transformation from the hierarchical competency data structure to list or context graphs is possible. However, the transformation from a list of competencies to a tree structure is not yet available. Text processing to a tree or tree-like structure is common [18], but it is oriented on word or phrase linking to a tree structure for sentence or concept generation. The tree of sentences is generated mostly in text clustering tasks [19] and not adapted to context area tree-like structuration. This complicates the automated integration of competency list-based e-learning systems to tree-based e-learning systems and requires additional attention.

3. Proposed Transformation from List of Competencies to Tree of Competencies

3.1. General Idea of the Transformation

Competencies mainly define the knowledge and skills of a student that one has (or should have) to successfully perform in professional, educational, and other life contexts. Commonly, competency is expressed as a verb, defining what the student will be able to do, in what the area, and the context in which this ability should be demonstrated.

Based on Bloom's taxonomy of the cognitive domain [20], the student's skills can be arranged into different levels: Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation. The learning should be organized for educating the lower level skills in the beginning and going to more complex skills just after lower-level skills are mastered. This meets the levels of the competency tree, where the lowest level skills are presented as leaves, while the most complex skills present the root of the tree. As the skill level in Bloom's taxonomy is defined by the verb, the verbs of the competency can be used to vertically align the competencies with the competency tree.

The most common verbs, defining each level of Bloom's taxonomy, is presented. Those could be used to define the level of competency. However, for some very specific disciplines, the verbs, defining the level of Bloom's taxonomy, might be very specific. To transform the competency list of very specific knowledge areas, the adjustment of common verbs for each Bloom's taxonomy should be done.

As a competency tree must be generated from the list of competencies, the vertical alignment is not enough, and the competencies have to be aligned horizontally, as well, to

get a tree structure. For this purpose, the context and area similarity should be taken into account. This is expressed mostly as nouns of the competency [21].

Combining the horizontal and vertical alignment of the competencies, the abstract architecture of the competency list transformation to a tree might be obtained (see Figure 1).

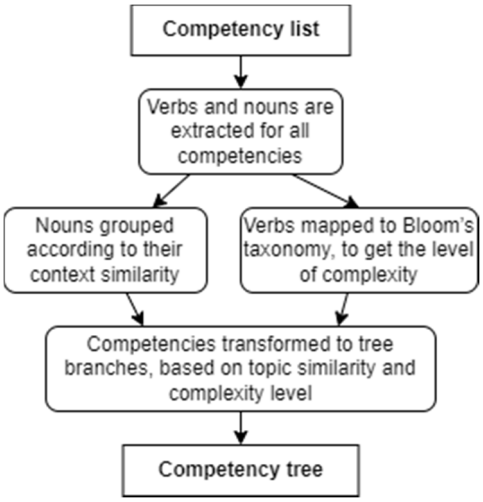


Figure 1. Abstract model of competency list transformation to competency tree.

In the transformation, all competencies should be analyzed to get the competency verbs and nouns, where verbs are used for vertical alignment of the competencies, while nouns analysis is used for context grouping. Each group of content should be inspected, and the parent of each group could be estimated based on the skill complexity. This would lead to the construction of tree branches. Consequently, each branch can be analyzed recursively, till the estimation of the tree root node.

The elements of the conceptual transformation model are presented in the following, to explain the algorithms behind it.

3.2. Competency Text Analysis for Noun and Verb Identification

For the identification of competency verbs and nouns, part of speech analysis (POS) is executed. Each sentence (S) is analyzed by using Python libraries for text parsing and tagging [22]. Based on sentence hierarchical structure (see Figures 2 and 3, where the processed sentence is presented at the bottom, while the POS tagging hierarchy is presented above each part of speech, word), the highest level verb phrase (VP) is selected for further analysis. The verb (VB) in it is selected as the main verb of the sentence, while the noun phrase (NP) is selected as the main content of the sentence. Additional prepositional phrases (PP) are analyzed to consider their inclusion into sentence verbs and nouns. Preposition phrases with joining prepositions, such as "by", "with", "to", "for", etc., are included for analysis, and the subsentence (lower level S) is analyzed in an analogue manner as the main one. The identified verbs and nouns are added to the list of sentence verbs and nouns (see Figure 2).

If the preposition in the sentence is not joining ("without", "except", etc.), nor are negative adverbs ("not", "no", etc.) used, the prepositional phrase is ignored and not included in the further analysis (see Figure 3).

The list of verbs is presented, not discriminating whether it was identified as the main verb in the prepositional phrase. All noun words are combined into one list, maintaining the order of words. This is done for further application or text analysis, to use bigrams, rather than unigrams only.

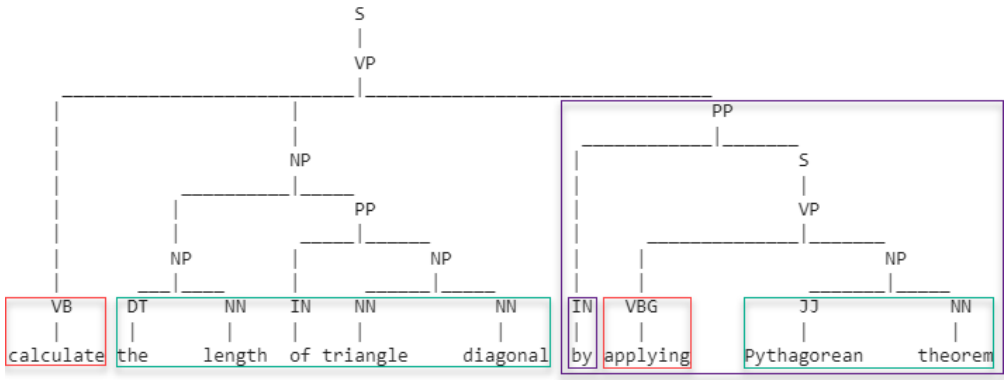


Figure 2. Example of competency text analysis, to get the skill verbs ("calculate" and "applying") and context, area of the competency ("triangle diagonal" and "Pythagoras theorem"), when joining preposition ("by") is used.

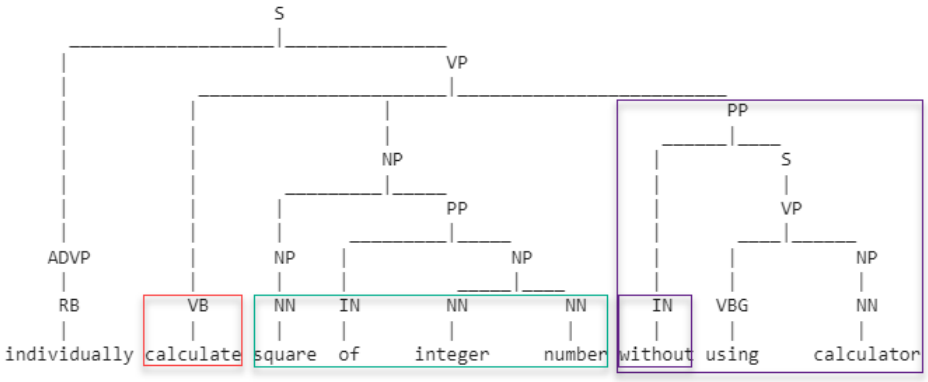


Figure 3. Example of competency text analysis, to get the skill verbs ("calculate") and context, area of the competency ("square" and "integer number"), when not joining preposition ("without") is used.

3.3. Context Grouping Implementation

The noun phrases in the competencies define the areas and context of the skill. Naturally, the branches of the competency tree should be constructed on the basis of the commodity of the child competencies and the variation of them. This leads to the tree structure being recovered by grouping similar competencies, based on its context. Therefore, the similarity between all competencies is analyzed by comparing its noun phrases only.

The number of noun phrases in each competency is very limited. This complicates the comparison, as the range of data for comparison is very limited. Therefore, text augmentation is used. The text pre-processing and augmentation is done in this order:

1. The tokenization of the noun phrases of a competence text has been performed. Tokenization is a way to separate text data into smaller units called tokens. In this research, tokens can be words, characters, punctuation signs, etc.
2. The tokens of competence are converted to lowercase.
3. All punctuation signs, numbers, and other non-important tokens have been removed.
4. For each word, synonymous words that express the same concept are obtained.
5. The lemma of each word and its synonyms is obtained and added to a list for each competence sentence.
6. Stop words and duplicated records are removed from the list, to leave only the main context of the competency text.

These steps modify the set of words that define the competency but do not change the context of the competence area significantly. Step 4 is impacted by the tools, used for context augmentation. For some specific disciplines, a database of area term synonyms might be developed. This requires long preparation; therefore, a general language corpus-based

synonym tool might be used. It potentially will not include some specific discipline terms, but the most common terms will be extended with possible synonyms.

For further similarity estimation, the composite list of terms is constructed by combining the word list of each of the competencies analyzed. This is the base of the term frequency-inverse document frequency (TF-IDF) vector. Based on this vector, each competence is expressed as a matrix, defining whether the words of the noun phrase appear in the vector or not.

With the TF-IDF matrix, the hierarchical/agglomerative clustering on the condensed distance matrix is estimated. This approach was selected to simplify the implementation of tree structure building, as the most similar records will be presented in near branches of the hierarchical clustering. However, hierarchical clustering does not build the needed tree structure, as the analyzed records are presented as leaves of the hierarchy tree, while each node in the competence tree should contain a competency.

To obtain the most suitable parameters, several methods for hierarchical clustering were performed, and cosine distance with Nearest Point Algorithm was selected for record analysis. The number of clusters is not necessary to be defined, as the input for other phases is the hierarchical structure and similar competencies, but it does not require getting a specific group of clusters.

3.4. Skill Complexity Estimation

The hierarchical clustering hierarchy defines the similarity between some records. At the same time, it created a new synthetic parent for each group of records. As the additional nodes are not needed, one of the child records should become a parent record. This record should cover the child records; therefore, the skill level, and complexity, should be higher. For each competency group, the competency with the highest difficulty level must be identified. This is done by estimating three parameters for each pair of competencies:

1. The difference in Bloom level, M_1—the difference between verb levels according to Bloom's taxonomy.
2. The difference in verb coverage, M_2—the maximum distance between hypernyms and hyponyms of verbs.
3. The difference in sentence coverage, M_3—the significant difference between the coverage of competence.

The first metric M_1 (difference of Bloom level) is estimated by comparing the competency verbs with the recommended verbs for each level of Bloom taxonomy [20]. When a competency has several verbs, the level is estimated by the verb with a higher Bloom taxonomy level. When comparing two competencies, the absolute value of the level is not important; the difference between levels in Bloom's taxonomy is estimated. If the difference is greater than 0, we assume that there is a difference, and the M_1 value for the competency with a higher level is assigned to 1; otherwise, it is equal to 0 (1).

$$M_1(A,\ B) = \begin{cases} 1, & max(\{Btl_A(v_1),\ \ldots, Btl_A(v_n)\}) - max(\{Btl_B(v_1),\ \ldots, Btl_B(v_n)\}) > 0 \\ 0, & max(\{Btl_A(v_1),\ \ldots, Btl_A(v_n)\}) - max(\{Btl_B(v_1),\ \ldots, Btl_B(v_n)\}) \leq 0 \end{cases}, \quad (1)$$

where $M_1(A,B)$ is a value for the difference of Bloom level for competency A, in comparison to competency B; $Btl_X(v_Y)$ is a Bloom taxonomy level for competency X, verb v_Y, where Y varies from 1 to n, and where n is the number of verbs in the competency.

The difference in verb coverage, M_2, does not rely on Bloom taxonomy but is dedicated to estimating the hypernyms, synonyms, and hyponyms of the verbs, as well as to estimating whether there is any difference. It is estimated only if the verbs in two compared competencies are different. In such a case, hypernyms, synonyms, and hyponyms for each verb in these two competencies are estimated. The hypernyms have a weight 1, and synonyms or the word have a weight 0, while the hyponym weight is -1. Such a verb list for each of the compared competencies is estimated, and matching verbs in these two lists are identified. If there are one or several matching verbs, the maximum difference between their weights is estimated. For the competency, with the higher verb level, the

M_2 value is assigned to 1, while, for the other one, it is assigned to 0 (2). This metric will represent which competency has a more general, wider range of verbs in the formulation of the competency.

$$M_2(A,\ B) = \begin{cases} 1, & max(\{mvl(mv_1),\ \ldots, mvl(mv_n)\}) > 0 \\ 0, & max(\{mvl(mv_1),\ \ldots, mvl(mv_n)\}) \leq 0 \end{cases}, \quad (2)$$

where M_2(*A*,*B*) is a value for different verb coverage for competency *A*, in comparison to competency *B*; *mvl*(mv_Y) is a weight difference between matching verb mv_Y of competence *A* and *B*, where *Y* is the *Y*-th matching verb out on *n* matches.

While M_1 and M_2 metrics are oriented on verbs only, the M_3 metric considers the full sentence. The difference in sentence coverage is estimated by estimating the similarity of two sentences. For each word in the sentence, synonyms are found. Synonyms of each sentence are compared pairwise, estimating the best distance between those words. For each word, in the first sentence, the best match score is found in the second sentence. The average of those scores is calculated as the score for the sentence. This kind of sentence similarity produces a different result if sentence *A* is compared to sentence *B*, and in the case when sentence *B* is compared to sentence *A*. Therefore, the value M_3 is assigned for the sentence with the higher score only if the difference between different directions is more than 15% (3). The difference indicates that one competency in its text has more terms in comparison to the other one and potentially covers a wider range of skills, or competencies.

$$M_3(A,\ B) = \begin{cases} 1, & sim(A,\ B) - sim(B,\ A) > 0.15 \\ 0, & sim(A,\ B) - sim(B,\ A) \leq 0.15 \end{cases}, \quad (3)$$

where M_3(*A*,*B*) is a value for different sentence coverage for competency *A*, in comparison to competency *B*; *sim*(*A*,*B*) is a similarity score between sentence *A* and sentence *B*.

When in a group of competencies, the parent competency needs to be identified, and all competencies are compared pairwise. In each comparison, the three scores are estimated (M_1, M_2, and M_3) for both of the competencies. In the end, all three scores are summed up to one sum *M* for each competency (4). The competency with the highest *M* value indicates the highest level competency and should be used as a parent for the rest of the competencies in the group.

$$M(A,\ B) = M_1(A,\ B) + M_2(A,\ B) + M_3(A,\ B), \quad (4)$$

where *M*(*A*,*B*) is a value of competency superior for competency *A*, against competency *B*.

3.5. Branch Forming and Adjustment

From the hierarchical clustering tree, the competency tree is formed recursively, analyzing all groups from the button to the top. For each node, the competency with the highest *M* value is assigned to be a parent node, while the rest of the competencies are assigned to be a child node. As the parent node is returned to the higher level of the hierarchical structure, it will be assumed as a node for analysis and one of the candidates to be a parent node. This forms a tree structure, where one competency will be assigned to be the root competency, while the rest competencies will form a hierarchical tree structure. The pseudocode of the free formation is presented in Algorithm 1.

However, this method has a limitation—when only two leaves are provided for analysis and parent identification, the node has only one child. This contradicts the principles of commodity and variation, as there is no sense to define a child node if it is the only parent of the node. To fix the limitation, some additional steps are implemented:

1. Fixing of single child situations by moving the single child to a higher-level node.
2. Fixing of merged nodes by indicating whether it needs to be split into a parent with multiple child nodes.

Algorithm 1. Pseudocode of tree formation from hierarchical clustering

```
1.  Input: node of the hierarchical clustering
2.  Output: branch of the competency tree
3.  formTreeNode(node)
4.      if node != None then
5.          if node is leaf then
6.              branch = new Branch (new Leaf(node text))
7.          else
8.              append formTreeNode(left branch of node) results to children list
9.              append formTreeNode(right branch of node) results to children list
10.             if size of children > 1 then
11.                 parent = getMaxLevelCompetency(children)
12.                 branch = new Branch (parent)
13.                 delete parent from children
14.                 add child competencies children to branch
15.             else
16.                 branch = new Branch (first children)
17.         end if
18.     end if
19. end if
```

The first fix starts from the bottom, leaves, and finds the nodes with one branch/leaf only. The branch/leaf will be moved one level up to eliminate the children in this node. As multiple branches with one child element in them can be merged into one node, it must be revised to inspect whether this node should be divided into several branches. Therefore, the second fix, first of all, inspects how many child nodes exist in the node. If there are more than 2 competencies, then the M values are estimated for each of the competencies. The list of child nodes is divided into a branch with one parent and multiple child competencies only if the M value of the highest level competency in the list has a higher value than half of the child count in it. If the condition is applicable, the highest-level competency is assigned to be the parent, while all the remaining competencies will be a child competency.

These two fixing functions are executed recursively, from the botton to the top, and ensure the commodity and variation rules are met.

4. Validation of the Transformation Model

To validate the proposed model, different examples of competency trees were analyzed. These examples covered different abstraction-level competency trees—some were chosen as task-level competency trees, while the other ones were course-level competency trees.

The evaluated competency trees were generated by different people or retrieved from e-learning courses. All those were taken from Lithuanian learning systems, covering examples in a range from primary school teaching plans to higher education study programs. To assure the quality of the competency tree, the test cases were obtained from persons (and their study courses) with experience in e-learning and expertise on hierarchy based (multi-level) competency evaluation. The test case trees were flattened to lists, leaving the competency text only, where, for each competency, the sentence was presented in a new line. The codes, or numbering, for each competency were generated; however, they were used just for visualization because codes were not used in the transformation model. All six examples of data and results of the transformation—generated tree—are presented in Appendix A, Table A1.

The competency trees were up to 15 nodes, and 3 levels in each. This matches the usual case, where the test should have no more than 15 competencies, to avoid overloading the student with too large a variety of topics or skills to cover. Therefore, in e-learning systems, for each activity, just portion, or a subtree, of the overall competency tree is used. Here, the selected examples reflect the practical size of the trees.

All test case competency lists were individually presented as input to the developed system for competency list automated transformation to competency tree. Each output of the transformation was compared to the initial competency trees. Only in 33% of the cases (cases no. 2 and 3) was the identical tree reconstructed from the competency list. Both cases included task level competencies in the field of math. There are not enough data to evaluate whether it is related to the topic; however, it might be affected by it, as these general topic terms are more often used, thus a wider variety of synonyms, hyponyms, and hypernyms existing in the language corpus.

Terminology in the competency description has a significant influence on tree transformation. For example, the terms "list" and "tree" are similar to the programmer, as defining types of data structure, while the similarity of these words is only up to 12%. Meanwhile, the similarity between the terms "addition" and "multiplication" reaches 29%. The lack of similarity between related terms identifies that the transformation could be improved by having a more specific lexical database. This could potentially increase the accuracy of hierarchical clustering.

Analyzing the transformation accuracy metrics, several metrics were analyzed.

1. Reference edges in the source. It defines the frequency of edges in the source (original) tree found in the target (generated) tree.
2. Source edges in reference. It defines the frequency of edges in the target that is (generated) tree found in the reference (original) tree.
3. Robinson-Foulds normalized distance [23] (normalized RF). It is calculated by dividing the Robinson-Foulds symmetric distance by the maximum Robinson-Foulds value for this comparison.
4. Correctly assigned nodes to branch. The number of nodes (competencies) that were assigned to a correct parent node.
5. Incorrectly assigned nodes to the branch. The number of nodes (competencies), whose parent is assigned incorrectly.
6. Accuracy. The proportion between the nodes correctly assigned to the branch and the total number of nodes in the competency list.
7. Distance resulting from the original tree. The minimal number of steps to move the appropriate node by one level, to get the original tree from the generated one.
8. List the distance from the original tree. The minimal number of steps to move the appropriate node by one level, to get the original tree from the initial competence list.

The first three metrics are commonly used for tree comparison. The normalized RF value in most of the cases is low, indicating the similarity between the tree structures. Only case no. 6 resulted in a clear difference between tree structures.

Another popular metric, TreeKO [24], is not suitable in this investigation, as all nodes are identical, just the edges are different. Therefore, the number of correctly assigned parents to the node was estimated. It is estimated by checking whether the node in the generated tree structure is a child of the same parent node as in the original tree. Accordingly, the proportion between correctly assigned nodes and the total number of nodes defines the accuracy of the transformation. In the six analyzed cases, the accuracy reached at least 50%, while the average accuracy was 72% (standard deviation is equal 22). It is not a perfect score, however, but indicates a promising application, as all six cases were evaluated by the same model, with no parameter adjustment. For practical application, different model parameters can be applied, and several outputs are proposed for the user to select the best one. However, it is worth mentioning that the competency lists for the experiment were selected from well-structured competency trees, where commodity and variation rules are applied. In real-world situations, some competency lists might lack some competencies to build a sequential tree. Therefore, additional solutions should be developed to estimate whether the list of competencies is suitable for transformation to tree or not.

The minimal distance to the original tree for the list and the generated tree is used to estimate the work needed to go from the current structure to the intended tree. Distance is

the number of nodes that move one level up or down the tree structure. As seen in Figure 4, the number of moves to obtain the original tree from the generated tree is significantly lower (p = 0.0028) compared to the initial list. The average distance is 4 times smaller in comparison to the list, which means that it would be easier to build the initial tree from the generated one, rather than building it from the competency list (the difficulty in understanding the content is not analyzed in this case).

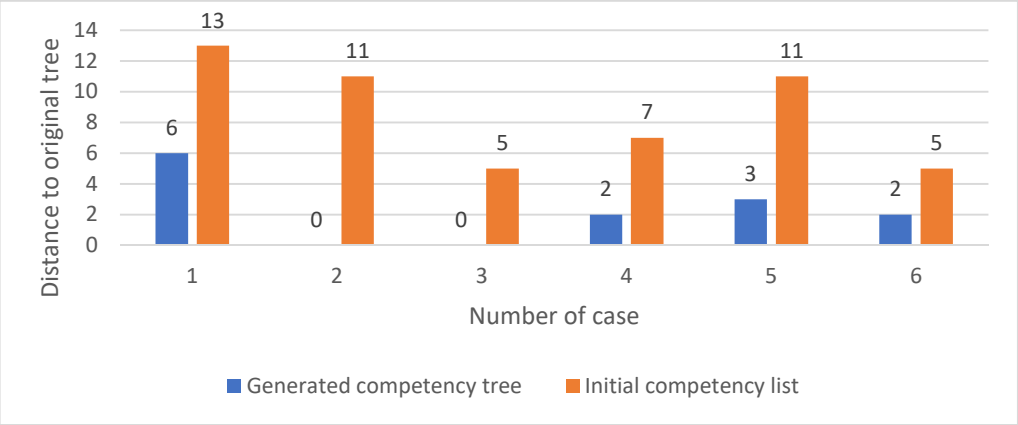


Figure 4. Comparison of distance, or steps, for node movement in the generated tree and in the initial competency list to get the full match to the original competency tree.

While the results show that the automated competency list transformation to tree is possible (two cases got an identical tree, while the rest of cases achieve not less than 50%accuracy), it is worth mentioning that, in the current state, it has several limitations. For example, this solution is only adapted to English. To adapt it for other languages, some additional text preparation and analysis methods should be used. At the same time, this solution is word-based; therefore, it is not resistant to spelling nor other language errors. In addition, the current version is adapted to a full competency tree only; only one tree will be generated, with all nodes with associated competency. For further development, some empty nodes or trees with multiple subtrees could be generated to cover this issue. The proposed transformation is promising as it helps to structure the competencies from a list into a tree structure, which is more promising in some e-learning and e-evaluation systems.

5. Conclusions

In e-learning and e-evaluation systems, different data structures are used for the storage of students' competency skills. All data structures have their advantages and application area; therefore, instead of standardizing all systems to use one solution, the transformation between different data structures might bring more benefits.

The generation of the competency tree from the competency list has its own specific rules, as some rules for the level and topic of the competency (commodity and variation) should be applied. The specifics allow tree generation from the list, which might be too complex for general cases, when any list of text sentences has to be presented as a meaningful tree.

The proposed model for the transformation of the competence list into the competence tree achieves 72% accuracy. This is not a very high result when taking into account that the standard deviation is 22%; however, it significantly reduces the number of steps needed to get the intended tree in comparison to the list. Therefore, it can be said that the current version has limitations for a fully automated competency list transformation to a competency tree; however, it can be used as a semi-automated tool, allowing reduction of human-expert work.

The current version is based on common English language corpus and verbs related to Bloom's taxonomy levels. This solution is adapted to the easier implementation of

language analysis and application for a wider range of disciplines. At the same time, it limits the transformation application in very specific disciplines, where specific terms, or abbreviations, are used and not presented in general-purpose language corpus. To improve it, each discipline should have a corpus, or language model, adapted to this specific discipline. This would require additional preparation for the model application but, potentially, would increase the accuracy of the transformations.

Author Contributions: Conceptualization, S.R. and A.M.; methodology, S.R. and P.S.; software, S.R.; validation, A.M., J.N. and P.S.; investigation, A.M.; writing—original draft preparation, A.M. and S.R.; writing—review and editing, A.M., S.R., J.N. and P.S.; visualization, A.M.; supervision, S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: To get the transformation Python code, contact the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Summary of all test cases and competency list transformation to tree results.

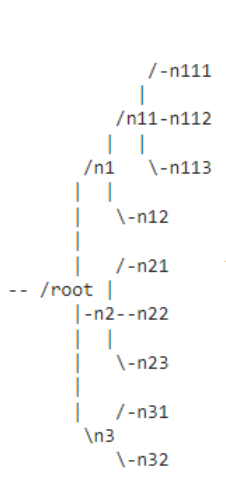
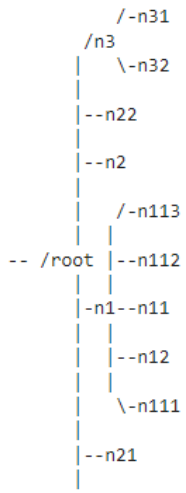
No.	Testing Data		Transformation Results		
	List of Competencies	Coding	Tree View	Tree View	Comparison Metrics
1	• Create simple applications. • Select the suitable data type for the number and string variables. • Assign a data type for the numeric variable. • Assign integer data type to numbers. • Assign float type to numbers. • Assign double type for numbers. • Assign string type to a variable. • Form a different output to the screen. • Output text symbol. • Output special symbol. • Output value of a variable. • Apply arithmetic and assignment operators. • Apply arithmetic operators. • Apply the main assignment operators.	• root • n1 • n11 • n111 • n112 • n113 • n12 • n2 • n21 • n22 • n23 • n3 • n31 • n32			reference edges in source = 1.0 source edges in reference = 0.5 normalized RF = 0.33 correctly assigned nodes to branch = 8 incorrectly assigned nodes to branch = 6 accuracy = 0.57 result distance to original tree = 6 list distance to original tree = 13

Table A1. *Cont.*

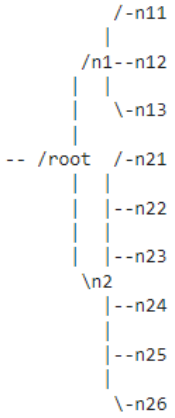
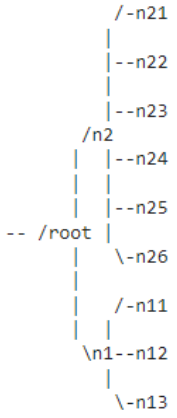
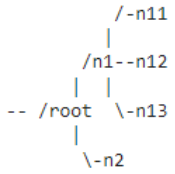
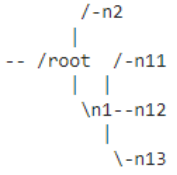
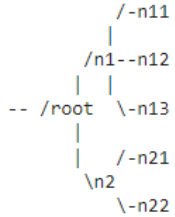
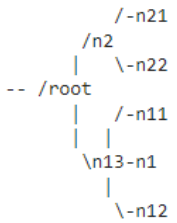
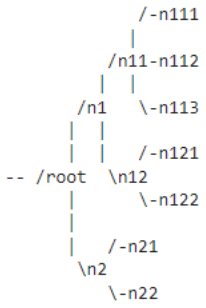
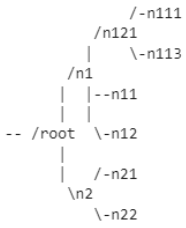
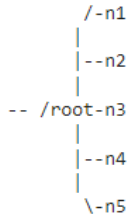
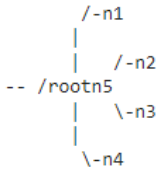
No.	Testing Data		Transformation Results		
	List of Competencies	Coding	Tree View	Tree View	Comparison Metrics
2	• Perform color conversion between different color space models. • Understand different color models. • Know the CMY color model. • Know the RGB color model. • Know the CMYK color model. • Understand the conversion of colours from one color model to another. • Convert the color from RGB to the CMY model. • Convert the color from CMY to RGB model. • Convert the color from CMY to CMYK model. • Convert the color from CMYK to the CMY model. • Convert the color from RGB to the CMYK model. • Convert the color from CMYK to the RGB model.	• root • n1 • n11 • n12 • n13 • n2 • n21 • n22 • n23 • n24 • n25 • n26			reference edges in source = 1.0 source edges in reference = 1.0 normalized RF = 0.0 correctly assigned nodes to branch = 12 incorrectly assigned nodes to branch = 0 accuracy = 1 result distance to original tree = 0 list distance to original tree = 11
3	• Apply mathematical knowledge to solve simple math tasks. • Understand the application principles of arithmetical operations for solving math tasks. • Know the principles of addition and subtraction. • Know the order of arithmetic operations. • Know the principles of multiplication and division. • Distinguish decimal numbers by understanding their values.	• root • n1 • n11 • n12 • n13 • n2			reference edges in source = 1.0 source edges in reference = 1.0 normalized RF = 0.0 correctly assigned nodes to branch = 6 incorrectly assigned nodes to branch = 0 accuracy = 1 result distance to original tree = 0 list distance to original tree = 5
4	• Design a computational algorithm for the solving of defined mathematical problems. • Understand the task presented in the written Lithuanian text. • Understand the meaning of verbs. • Understand the meaning of nouns. • Understand the meaning of numbers presented in a text. • Solve arithmetic math tasks, including addition, subtraction, multiplication, and division operations are used. • Use addition and subtraction operations. • Multiplication and division operations.	• root • n1 • n11 • n12 • n13 • n2 • n21 • n22			reference edges in source = 1.0 source edges in reference = 1.0 normalized RF = 0.0 correctly assigned nodes to branch = 4 incorrectly assigned nodes to branch = 4 accuracy = 0.5 result distance to original tree = 2 list distance to original tree = 7

Table A1. *Cont.*

No.	Testing Data		Transformation Results		
	List of Competencies	Coding	Tree View	Tree View	Comparison Metrics
5	• Design a qualitative software architecture by applying suitable algorithms and data structures. • Select the data structure for the required architecture implementation. • Apply the principles of linear data structures. • Know the principles of queue structure. • Know the principles of heap structure. • Know the principles of the list structure. • Apply the architecture of hierarchical data structure. • Understand the implementation of the graph structure. • Understand the implementation of the tree structure. • Implement the principles of various data processing algorithms in the developed software. • Understand the principles of various search algorithms. • Understand the principles of various sorting algorithms.	• root • n1 • n11 • n111 • n112 • n113 • n12 • n121 • n122 • n2 • n21 • n22			reference edges in source = 1.0 source edges in reference = 1.0 normalized RF = 0.0 correctly assigned nodes to branch = 7 incorrectly assigned nodes to branch = 5 accuracy = 0.58 result distance to original tree = 3 list distance to original tree = 11
6	• Implement solutions for specific communication campaigns, creatively integrating art applications, taking into account the needs of different audiences and target groups. • Organize public relations activities from the perspectives of individual, organizational, and mass communication. • Plan the impact of information on certain groups of society. • Prepare information for dedicated society groups. • Evaluate the results achieved by linking them with certain groups or organizations in society. • Select appropriate communication channels for the specific group and dedicated information.	• root • n1 • n2 • n3 • n4 • n5			reference edges in source = 0.0 source edges in reference = NA normalized RF = 1.0 correctly assigned nodes to branch = 4 incorrectly assigned nodes to branch = 2 accuracy = 0.67 result distance to original tree = 2 list distance to original tree = 5

References

1. Mouratidis, K.; Papagiannakis, A. COVID-19, internet, and mobility: The rise of telework, telehealth, e-learning, and e-shopping. *Sustain. Cities Soc.* **2021**, *74*, 103182. [CrossRef] [PubMed]
2. Margiene, A.; Ramanauskaite, S. Automated E-Assessment: Students' Needs and E-Evaluation Solution Possibilities. Available online: http://www.ijiet.org/show-157-1913-1.html (accessed on 30 December 2021).
3. Birjali, M.; Beni-Hssane, A.; Erritali, M. A novel adaptive e-learning model based on Big Data by using competence-based knowledge and social learner activities. *Appl. Soft Comput.* **2018**, *69*, 14–32. [CrossRef]

4. Hatzilygeroudis, I.; Koutsojannis, C.; Papachristou, N. Adding adaptive assessment capabilities to an e-learning system. In Proceedings of the 2006 First International Workshop on Semantic Media Adaptation and Personalization (SMAP'06), Athens, Greece, 26 December 2006; pp. 68–73. [CrossRef]
5. Chatzopoulou, D.I.; Economides, A.A. Adaptive assessment of student's knowledge in programming courses. *J. Comput. Assist. Learn.* **2010**, *26*, 258–269. [CrossRef]
6. Zlatović, M.; Balaban, I.; Hutinski, Ž. A Model of the Continual Adaptive Online Knowledge Assessment System. In *E-Learning and Digital Education in the Twenty-First Century-Challenges and Prospects*; IntechOpen: London, UK, 2020. [CrossRef]
7. Aukstakalnis, N.; Baniulis, K.; Pauliute, J.; Slotkiene, A. Graphical model: The means for simulation-based learning. In Proceedings of the ITI 2008—30th International conference on Information Technology Interfaces, Cavtat, Croatia, 23–26 June 2008; Volume 16, pp. 471–476.
8. García, K.; Brézillon, P. Model visualization: Combining context-based graph and tree representations. *Expert Syst. Appl.* **2018**, *99*, 103–114. [CrossRef]
9. Fonseca, A.; Faria, H. Adaptive knowledge assessment using advanced concept maps with logic branching multiple-choice Google Forms. *eLearn* **2021**, *2021*, 1–12. [CrossRef]
10. Silva, A.; Padilha, N.; Siqueira, S.; Revoredo, K.; Baião, F. Using Concept Maps and Ontology Alignment for Learning Assessment. *IEEE Technol. Eng. Educ.* **2012**, *7*, 33–40.
11. Ramanauskaite, S.; Slotkiene, A. Hierarchy-Based Competency Structure and Its Application in E-Evaluation. *Appl. Sci.* **2019**, *9*, 3478. [CrossRef]
12. Ausubel, D.P.; Novak, J.D.; Hanesian, H. *Educational Psychology: A Cognitive View*, 2nd ed.; Holt, Rinehart and Winston: New York, NY, USA, 1978.
13. Kinchin, I.M. *Visualizing Powerful Knowledge to Develop the Expert Student: A Knowledge Structures Perspective on Teaching and Learning at University*, 1st ed.; Sense Publishers: Rotterdam, The Netherlands, 2016; pp. 15–73.
14. Novak, J.D.; Gowin, D.B. *Learning How to Learn*; Cambridge University Press: New York, NY, USA, 1984.
15. Novak, J.D.; Cañas, A.J. *The Theory Underlying Concept Maps and How to Construct and Use Them*; Technical Report IHMC CmapTools; Florida Institute for Human and Machine Cognition: Pensacola, FL, USA, 2008. Available online: http://cmap.ihmc.us/docs/pdf/TheoryUnderlyingConceptMaps.pdf (accessed on 30 December 2021).
16. Liang, L.; Deng, X.; Liu, Q. Task-Driven and Objective-Oriented Hierarchical Education Method: A Case Study in Linux Curriculum. In Proceedings of the 2008 IEEE International Symposium on IT in Medicine and Education, Xiamen, China, 12–14 December 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 316–318.
17. Margienė, A.; Ramanauskaitė, S. Toward Adaptability of E-Evaluation: Transformation from Tree-Based to Graph-Based Structure. *Appl. Sci.* **2021**, *11*, 4082. [CrossRef]
18. Bai, Y.; Li, Z.; Ding, N.; Shen, Y.; Zheng, H.-T. Infobox-to-text generation with tree-like planning based attention network. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021.
19. Zhang, H.; Wang, C.; Wang, Z.; Duan, Z.; Chen, B.; Zhou, M.; Henao, R.; Carin, L. Learning Hierarchical Document Graphs From Multilevel Sentence Relations. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *early view*. [CrossRef] [PubMed]
20. Huitt, W. Bloom et al.'s taxonomy of the cognitive domain. *Educ. Psychol. Interact.* **2011**, 1–4. Available online: http://www.edpsycinteractive.org/topics/cognition/bloom.pdf (accessed on 30 December 2021).
21. Minguillón, J.; Conesa, J.; Rodríguez, M.E.; Santanach, F. Learning analytics in practice: Providing E-learning researchers and practitioners with activity data. In *Frontiers of Cyberlearning*; Springer: Singapore, 2018; pp. 145–167.
22. Bird, S.; Loper, E. NLTK: The natural language toolkit. In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, Philadelphia, PA, USA, 7 July 2002. [CrossRef]
23. Pattengale, N.D.; Gottlieb, E.J.; Moret, B.M.E. Efficiently computing the Robinson-Foulds metric. *J. Comput. Biol.* **2007**, *14*, 724–735. [CrossRef] [PubMed]
24. Marcet-Houben, M.; Gabaldón, T. TreeKO: A duplication-aware algorithm for the comparison of phylogenetic trees. *Nucleic Acids Res.* **2011**, *39*, e66. [CrossRef] [PubMed]