*proceedings*

MDPI

*Proceedings*

# Architecture for Efficient String Dictionaries in E-Learning †

**Antonio Ferrández [1,*], Jesús Peral [1], Higinio Mora [2] and David Gil [2]**

[1] Department of Software and Computing Systems, University of Alicante, 03080 Alicante, Spain; jperal@dlsi.ua.es
[2] Department of Computing Technology and Data Processing, University of Alicante, 03080 Alicante, Spain; hmora@ua.es (H.M.); david.gil@ua.es (D.G.)
* Correspondence: antonio@dlsi.ua.es; Tel.: +34-96-590-3400
† Presented at the 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018), Punta Cana, Dominican Republic, 4–7 December 2018.

Published: 18 October 2018

**Abstract:** E-Learning is a response to the new educational needs of society and an important development in Information and Communication Technologies. However, this trend presents many challenges, such as the lack of an architecture that allows a unified management of heterogeneous string dictionaries required by all the users of e-learning environments, which we face in this paper. We mean the string dictionaries needed in information retrieval, content development, "key performance indicators" generation and course management applications. As an example, our approach can deal with different indexation dictionaries required by the course contents and the different online forums that generate a huge number of messages with an unordered structure and a great variety of topics. Our architecture will generate an only dictionary that is shared by all the stakeholders involved in the e-learning process.

**Keywords:** string dictionaries; e-learning architecture; context awareness in learning process

## 1. Introduction and Motivation

The methods of learning have experimented a tremendous change, especially due to the novelties in the technology as well as the requirements requested by society. We can highlight the advent of the online frameworks in their diverse ways, Open and Distance Learning (ODL) [1], as the key factor. The inherent features of these sites offer all types of educational tools to many students around the world. Hence, a very significant change in the educational costs can be established from the more concrete elements (such as educational buildings) to the technological infrastructures that provide knowledge.

The ODL platforms are mainly implemented as online web frameworks for education and therefore, they provide all the facilities for integrating the services used in the educational environment. These tools allow collaboration among all participants improving interaction. The increasing of the collaborative framework based on the web allows users to share information and take advantage of the interactions of other users. This fact leads to enhance their experience providing numerous benefits such as an increase in student motivation and the creation of collective intelligence [2]. This continuous development of web possibilities enhances the teaching-learning process and increases the effectiveness of learning systems in the knowledge society [3]. However, there are still challenges that need to be addressed for effective ODL provision and to maintain excellent educational standards.

The online forum represents one of the most powerful and popular tools [4] and is a communication technology tool frequently used in education. Online forums provide an excellent

platform for learning and connecting students to the subject. They increase student engagement in the subject, promote deep learning, and maintain motivation [5]. However, the challenge they present is related to the task of managing the huge number of messages that are generated. This can result in topics becoming fragmented over many threads with no search facilities to discover relevant information [5].

The search and information management roles may be enhanced by technological advances and computing methods that facilitate the educational process. Thus, Artificial Intelligence (AI) approaches based on Natural Language Processing (NLP) can deliver tools specifically aimed at students and teachers. The objective would be to make the assimilation of content and course tracking easier for students. Furthermore, the tools would be designed to manage the learning platforms by teachers, especially in online courses (MOOCs and those based on collaborative learning [6]).

The working hypothesis of our research is that this kind of solutions (AI approaches based on NLP), used to manage ODL learning platforms by teachers and students, require complex architectures that must deal with huge textual corpora. For example, Information Retrieval and Question Answering tools will require heterogeneous string dictionaries to handle with searches on the course contents and the different online forums that generate a huge number of messages with an unordered structure and a great variety of topics. These data sets can grow to become so large and complex that they are difficult to process using traditional data processing applications. Moreover, modern NLP algorithms are often based on machine learning techniques that learn through the analysis of large corpora of real-world examples. For example, statistical machine translation systems require parallel corpora formed by millions of sentence pairs, which are often too large to fit entirely into main memory [7]. Therefore, in this paper we propose an architecture that allows a unified management of heterogeneous string dictionaries required by all the users of e-learning environments with high benefits resulting in a reduction of the required memory and the guaranteed interactions between all the textual resources (it improves the Human-Computer interaction supporting a global and specific data mining of the textual corpora).

The paper is structured as follows. In Section 2, we summarize the most relevant work related to our proposal. In Section 3, we introduce our architecture. In Section 4, we experimentally analyze its efficiency. We conclude the paper with a summary of the main contributions and the directions for future work.

## 2. Related Work

Because of the extensive work that has been done in this area, this section has been divided into two parts. The first one will deal with the work related to the management of e-learning environments. The second one will summarize the most relevant data structures for dealing with string dictionaries.

### *2.1. Management of E-Learning Environments*

As previously mentioned, e-learning platforms deal with teachers, students and managers of the courses. Each user of these platforms presents different kinds of necessities. For example, the managers and teachers will require to obtain performance indicators [8] with the aim of managing online courses and designing efficient learning strategies, such as the analysis of teacher and student participation rates, the extraction of qualitative and quantitative measures (student and performance motivation, dropout rate, etc.).

Regarding to the diverse search operations needed by teachers and students, they can affect to both the contents of the course and the online forums generated by teachers and students. Community Question Answering (CQA) research line covers much work related to searches on question-and-answer websites such as Stack Overflow or Ask Ubuntu (a similar context to the one in the online e-learning forums). These searches present an especial treatment, since they will require diverse actions [9]: (1) to find the most similar question to the original question; (2) to find the most likely answer; (3) to determine if a question is new (it does not exist in the forum) to find the correct

answer; (4) to determine if a question is similar to an existing question, in which case the most likely answer should be found.

The CQA forums are increasingly gaining popularity. Consequently, since 2015 during a series of ongoing annual competitions SemEval—Semantic Evaluation—(http://en.wikipedia.org/wiki/SemEval), one of the tasks that has been developed is denominated Track 3 (http://alt.qcri.org/semeval2017/task3/). Track 3 evaluates systems that carry out the automatic process of finding good answers to new questions in a forum of discussion created by the community. For example, by retrieving similar questions in the forum, the correct answers can be identified. In the SemEval-2017 competition, two additional subtasks to Track 3 have been proposed. In the first subtask, given a new question and a set of related questions from the collection, similar questions needed to be classified according to their similarity to the original new question (with the idea that the answers to similar questions must answer the original question also). In the second subtask, given a question from a question-answer thread, all the response posts were classified according to their relevance to the question. A detailed description of the Semeval 2016 and 2017 editions and their participants are presented in [10,11].

### *2.2. Data Structures for Dealing with String Dictionaries*

Typical data structures for textual indexing are Hash Tables and some variants of Tries. The *Trie*, also called *Digital Tree* or *Prefix Tree* [12,13], is one of the main data structures used for indexing text. It is a tree for storing strings in which there is one node for every common prefix and the strings are stored in extra leaf nodes. Unlike a *Search Tree*, the nodes in the Trie do not represent elements stored in the structure, but the join of the characters of the nodes that form the path from the root to the node represent the string stored [14]. The Trie is a rather compact structure because overlapping prefixes are stored only once. It resolves a query that tests whether a given string belongs to the Trie in time only proportional to its length, unlike other data structures such as *Search Trees* or *Hash Tables* (an associative array whose keys are mapped to array positions by hash functions) in which the search time is a function of the number of elements stored. Therefore, the Trie search operations are faster because in most dictionaries the number of words is higher than the maximum length of a word. Moreover, Tries do not require complex re-balancing or tree rotation operations. Furthermore, Tries can be used to represent full-text indexes too: for example, the *Suffix Trie* is a Trie that stores all the suffixes of a given string.

The main disadvantage of Tries is the high memory consumption, especially when the set of words is heterogeneous with a low degree of prefix sharing. This drawback is partly solved by its compact representation called *Radix Tree*, *Patricia Trie*, *Compact Prefix Tree* or *C-Trie*, where all nodes with one child are merged with their parents. In [15], a dynamic construction algorithm for the Compact Patricia Trie is proposed to solve the problem that the Patricia Tree requires many good physician storage spaces in memory, especially when the key set is too large to fit in memory.

In addition to the compaction operation, another operation to improve the memory consumption in Tries is the minimization. The resulting structure is called *Directed Acyclic Word Graph (DAWG)* or *Acyclic Deterministic Finite Automata (ADFA)*, where unlike compaction, each edge between nodes is labelled with a character and consequently the number of nodes is significantly reduced. Essentially, a DAWG/ADFA is a finite state machine that recognizes a set of words [16]. When both compaction and minimization are performed, the *Compact Directed Acyclic Word Graphs (CDAWGs)* are obtained [17].

## 3. The Proposed Architecture

In this section, an architecture is presented that supports a unified management of heterogeneous string dictionaries required by all the users of e-learning environments. It can deal with different indexation dictionaries required by the course contents and the different online forums that generate a huge number of messages with an unordered structure and a great variety of topics. These dictionaries provide the required information by all the users of these e-learning

platforms such as "key performance indicators" (KPI) generation or generic searches on all the textual information generated in the platform.

Our architecture differs from previous work because it will generate an only dictionary that is shared by all the stakeholders involved in the e-learning platform. In this way, a high reduction of the spatial requirements for storing the dictionaries is achieved. Another benefit in our proposal is that the interrelations between the different dictionaries is provided automatically by the shared dictionary, because it supports KPI extraction grouped by different dictionaries or group of dictionaries (e.g., the KPIs can be generated for an specific or all the online forums). The shared dictionary will be implemented as a Trie, because it is based on the hypothesis that Tries are more space efficient as they store more elements that share prefixes, as well as it does not increase the temporal complexity in the operations, given that this data structure keeps the complexity of the insertion, search, modification and deletion operations proportional to the length of the string. Additionally, it is based on the fact that computational applications that require text processing usually need more than one textual resources (e.g., an Information Retrieval application can require a semantic dictionary, a lexical dictionary, lists of stop-words, etc.), which are habitually stored in separated data structure indexes.

This proposal presents an additional contribution in order to represent relations between different words, as it is required in n-grams, phrases, bitexts or biwords. Other authors (e.g., [18–20]) store these resources by assigning codes to words or shared substrings between different terms [21]. Our architecture automatically provides these codes or word identifiers. For example, if the following word identifiers are automatically assigned for the words: hat (6) and mat (12). Therefore, we can store easily the bi-gram frequency of the pair "hat-mat" by indexing the identifiers "6–12" in a separated data structure.

The proposed shared Trie is represented in one-dimensional arrays unlike the traditional pointer-linked structure of Radix Trees and Tries: memory pointers are substituted by offsets for the links between nodes. It is especially important on 64-bit machines, in which pointers require 8 bytes. Thanks to this sequential representation, after the Trie is created, when it is stored in secondary memory, the subsequent load times are significantly reduced because the dynamic reservation of the array memory is performed in one single operation. Loading each array position is more efficient than the link representation, which requires creating and linking each node individually. Moreover, it would allow a direct access to the structure when it is stored in secondary memory in case that it does not fit in main memory due to the number of keys to be indexed.

Our structure is totally dynamic by using a *primary array* of a fixed size BN0 and a *list of arrays* (each one of size BN). This list is populated once the primary array is full. Each array position is indexed sequentially starting from zero (primaryArray[0]) in order to stablish descendent, sibling and parent relations between Trie nodes. The indexes of Trie nodes stored in the list of arrays continues after the last index as presented in Figure 1.b (e.g., index 8 stores the Trie node with letter "l", and with its first descendent in index 10: the node with letter "d").

The way to calculate the array position for a Trie index is the following. Let "p" be an Trie node index that ranges from zero to the number of occupied array positions (e.g., in Figure 1.b, the maximum "p" is 11). If "p" is lower than BN0, then it is accessed as "primaryArray[p]". Whereas, if it is upper than BN0, then it is located in the "list of arrays" node number obtained from the integer division: "(p—BN0)/BN" (the first "list of arrays" node is numbered as zero). In this list node, it is stored in the array position "(p—BN0) MODULUS BN" (so the value returned will be a range between 0 and BN-1, because the numeration in each array ranges between these values). For example, in Figure 1.b with BN0 = 7 and BN = 3, index 11 is higher than BN0, then it is located in the list of arrays. It is in the list node number (11−7)/3 = 1. In this list node, it is stored in the array position [(11–7) MODULUS 3 = 1], which contains the value "par: 8" that means that the parent of the Trie node stored in index 11 is in index 8.

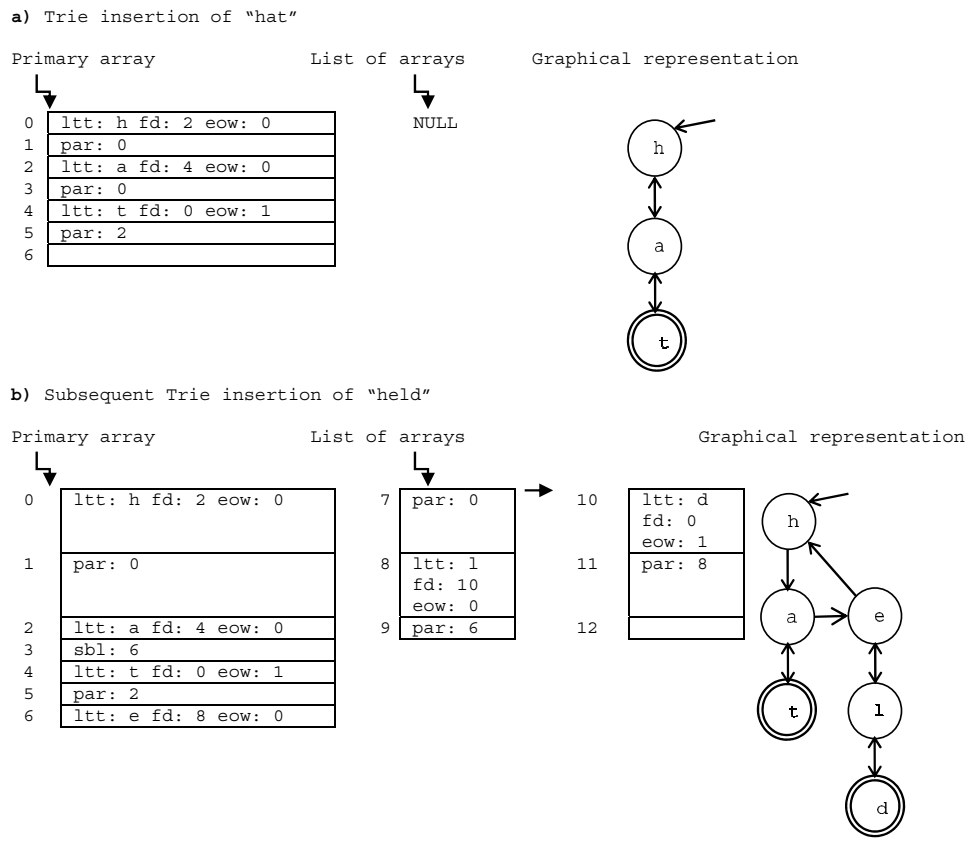


**Figure 1.** Sequential and graphical representations of Trie (ltt: letter; fd: first descendent index; eow: end of word; par: parent index; sbl: sibling index; BN0 = 7; BN = 3) that indexes the terms *hat* and *held*.

With the aim of clarifying the intuitive structure of Tries, Figure 1 illustrates two Trie examples, in their sequential (the primary and list of arrays) and graphical representations (the tree structure depicted on the right hand side of the Figure). The graphical representation shows the links stored in the sequential representation, exposing the Trie as a Trie structure that allows both upward (with the "par" index) and downward traversals (with the "fd" index). The set of descendants of a node are represented as an unsorted list: the first descendent is linked through the "fd" field, the end of the list is marked when the "par" parent index appears, and each new sibling node is inserted at the end of the list through the "sbl" sibling index. For example, the root of the Trie in Figure 1.b is stored in index 0: "ltt: h fd: 2 eow: 0", so the letter in the node is "h", its first descendent is in index 2, and it does not mark the end of a word. This descendent (ltt: a fd: 4 eow: 0) is followed by index 3 that contains its "sbl" sibling index (6): "ltt: e fd: 8 eow: 0", which is followed by index 7 with the "par: 0" parent index that marks the end of the list of sibling nodes, and also represents the parent of this node (the root with letter "h"). This list is depicted in the second level of the graphical representation.

The Trie insertion and search algorithm is similar to the one used in traditional Tries. In Figure 1.a, the word "hat" is inserted, which supposes that six array positions are required: two per node. For example, the primaryArray[4] = "ltt: t fd: 0 eow: 1" is followed by the correspondent parent index, primaryArray[5] = "par: 2", which means that the parent of the node with letter "t" is in index 2: the node with letter "a". Let us suppose that the word "held" is inserted subsequently, whose result is depicted in Figure 1.b. It requires additional memory that is stored in two "list of arrays" nodes (each one is an array of size BN = 3). This new word insertion implies that the node with letter "a" has a new sibling node with letter "e", so the primaryArray[3] changes from the previous value of "par: 0" to "sibl: 6", the position where the new node is inserted, which is followed by the primaryArray[7] = "par: 0" (i.e., the node with letter "h"). In the part b) of this Figure, we can see that only two nodes have the "end of word" flag set to 1: positions 4 and 10, which mark the end of the

two words inserted in the Trie. This eow flag is depicted in the graphical representation as the nodes with doubled line and will provide the unique identifiers of each indexed term.

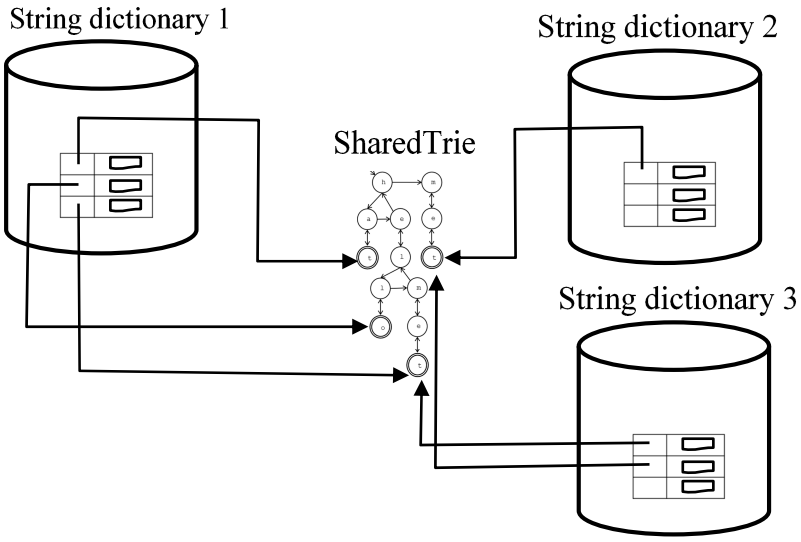


**Figure 2.** The proposed architecture.

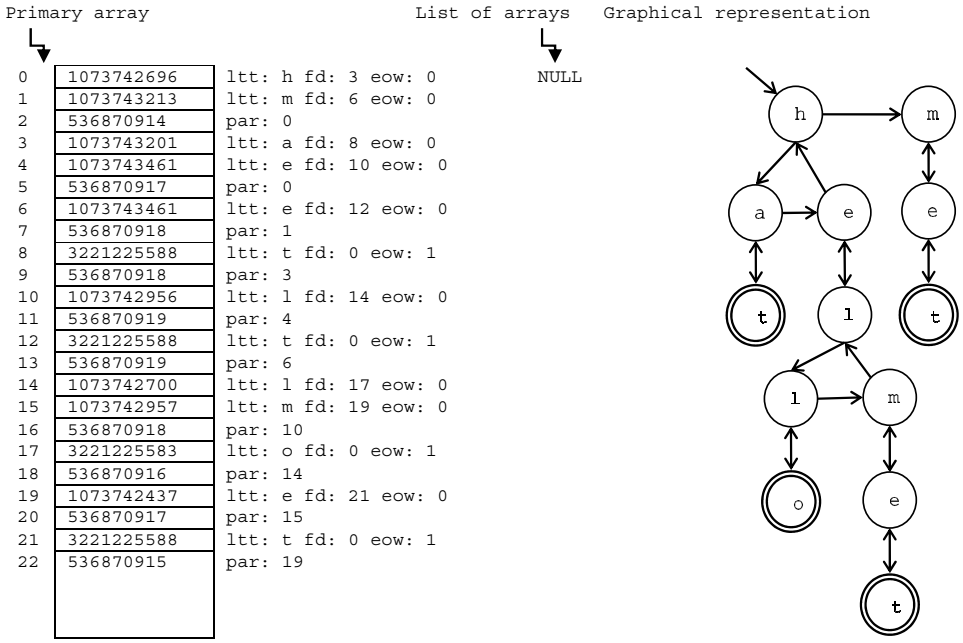


**Figure 3.** Trie with terms and identifiers: hat (8), hello (17), helmet (21) and met (12). (ltt: letter; fd: first descendent index; eow: end of word; par: parent index; sbl: sibling index; BN0 = 23; BN = 5).

The interrelations between the shared Trie and each dictionary in the proposed architecture is depicted in Figure 2. We propose that all these textual resources are stored in a single structure (from now on SharedTrie) with the arrays previously described. Moreover, the indexes of the Letter Nodes with the "end of word" bit set to 1 (i.e., the nodes that mean the end of a word in the Trie) will be used as unique identifiers of each word. The identifiers of the words of each text resource are stored in independent Hash Tables, which also store the data associated to each word. Therefore, the advantages of Hash Tables are profited in Tries: efficient looking-up data with low space complexity. In this way, the number of Trie nodes is highly reduced since different text resources share them. For example, let us suppose that the SharedTrie is the one presented in Figure 3 formed by the words with their respective identifiers: hat (8), hello (17), helmet (21) and met (12); and there

are two "StringDictionary a, b", the first one with the words "hat, helmet" and the second one with "hello, met". Then, each StringDictionary will have its own Hash Table. "a" will contain a Hash Table with the identifiers 8 and 21, whereas "b" will contain 17 and 12. Moreover, the insertion in "b" of the word "helmet" will be highly efficient because it does not suppose any node insertion since the word is previously inserted in the SharedTrie. Therefore, we only have to insert the identifier 21 in the Hash Table of "b".

## 4. Experimental Evaluation

The experiments have been run on a 64 bit 8x Intel(R) Xeon(R) CPU E5606—2.13 GHz—32 Gb RAM computer with Ubuntu14.04 LTS (Linux 3.13.0-95-generic SMP x86_64 GNU/Linux). Our C++ implementation has been compiled with gcc version 4.8.4 with the options "-std = gnu++0x-O3". The dataset used in the experiments are detailed in Table 1 and consist of 35,094,468 bytes and 5,383,339 words, where the first six files corresponds to WordNet lexical database, the following two files corresponds to standard stopword files, the penultimate file is a dictionary with morphological knowledge, and the last one is a corpus of forums extracted from the free online course named titled "Statistics in Medicine" https://lagunita.stanford.edu/courses/Medicine/MedStats./Summer2015. This set of files is usually required in a typical e-learning platform. In this way, the proposed experiments emulate a real situation where all this textual information must be processed and indexed in the same platform.

**Table 1.** Description of the datasets used in the experiments.

| Bytes | File | Num. Words |
|---|---|---|
| 832,275 | Dictionary of noun synonyms | 74,487 |
| 153,215 | Dictionary of verb synonyms | 14,585 |
| 1,154,098 | Dictionary of noun hyperonyms | 121,484 |
| 732,253 | Dictionary of noun hyponyms | 74,570 |
| 205,808 | Dictionary of verb hyperonyms | 21,664 |
| 134,413 | Dictionary of verb hyponyms | 13,733 |
| 2,152 | Spanish Stopwords | 344 |
| 2,485 | English Stopwords | 429 |
| 2,136,728 | Lexical dictionary | 243,297 |
| 29,741,041 | Corpus of online forums | 4,818,746 |
| **35,094,468** | **TOTAL** | **5,383,339** |

The experiment is run on the whole dataset in Table 1, trying to prove that our proposal is not degraded as the number of dictionaries grows: 5,383,339 words and 10 dictionaries, with no temporal complexity increase in search and insert operations. The results are summed up in Table 2, in which the memory consumption has been measured by using the C++ program in http://dis.um.es/~ginesgm/files/doc/memory.cpp; http://dis.um.es/~ginesgm/medidas.html#mt that returns the maximum memory usage in kilobytes by scanning the file "/proc/process_id/smaps" 16 times per second. The running time shown has been measured in seconds by the getrusage function that reports resource usage totals for processes. The "Trie" and "Radix Tree" row in Table 2 denotes our C++ Trie and Radix Tree implementation. Finally, the "Hash Table" indicates the STL Hash Table "unordered_set<string>", in order to perform an external evaluation and comparison with an efficient and widely used data structure. The SharedTrie is the most memory efficient, followed by the Radix Tree in 37%. The Hash Table presents the best temporal efficiency, but with no significative differences regarding our proposal (the second fastest one): 4.5 s in comparison with the 2.5 s employed in 5,383,339 search operations (the whole corpora). Moreover, the time required for the creation operation is much lower in the SharedTrie in comparison with the Trie, Hash Table and Radix Tree: 0.5 s, 7.05 s, 18.6 s, 3.4 s and 158.5 s respectively, which proves the benefits of our proposal. The highest creation time measured for the Radix Tree corroborates the fact that it has a significant cost when compressed segments need to be split or merged.

**Table 2.** Results of the experiment.

| | (b)Max. memory usage (kB) | | Search Time (s) | |
|---|---|---|---|---|
| SharedTrie | 32,632 | | 4.5 | |
| Trie | 85,588 | +162% | 23.6 | +421% |
| Hash Table | 60,876 | +87% | 2.5 | -44% |
| Radix Tree | 44,828 | +37% | 61.6 | +1,259% |

## 5. Conclusions and Future Research

In this paper, we have faced a problem present in many e-learning platforms about the management of all the textual information generated by these platforms. We have proposed an architecture that supports a unified management of heterogeneous string dictionaries required by all the users of e-learning environments. Our architecture will generate an only dictionary that is shared by all the dictionaries in the e-learning courses. The main advantages of our proposal are: (1) the reduction of the spatial requirements for storing the dictionaries; (2) the interactions between all the dictionaries that are supported automatically (e.g., can be used to obtain key performance indicators from the shared dictionary for the whole course management or an specific forum, which results in an improved Human-Computer interaction, both global and specific). The benefits of our architecture are experimentally proved in comparison with other proposals. As future projects, the authors plan to release the architecture as a public C++ library. Moreover, we will try to improve the temporal efficiency with alternative representations and implementations. This fact will aim us to explore other domains to apply, and especially in the health area where the amount as well as the variety of data is huge.

**Acknowledgments:** This work was supported in part by the Spanish Ministry of Economy and Competitiveness (MINECO) under Project SEQUOIA-UA (TIN2015-63502-C3-3-R), Project RESCATA (TIN2015-65100-R) and Project PROMETEO/2018/089; and in part by the Spanish Research Agency (AEI) and the European Regional Development Fund (FEDER) under Project CloudDriver4Industry (TIN2017-89266-R).

## References

1. UNESCO. *Open and Distance Learning. Trends, Policy and Strategy Considerations*; UNESCO: Place de Fontenoy, Paris, 2002.
2. De Azevedo, V.L.L.; Borges, M. More collaboration, more collective intelligence. *Int. J. Knowl. Soc. Res.* **2015**, *6*, 1–18.
3. Lytras, M.D.; Mathkour, H.I.; Abdalla, H.; Al-Halabi, W.; Yanez-Marquez, C.; Siqueira, S.W.M. An emerging—Social and emerging computing enabled philosophical paradigm for collaborative learning systems: Toward high effective next generation learning systems for the knowledge society. *Comput. Hum. Behav.* **2015**, *51*, 557–561.
4. Mora H.; Ferrández, A.; Gil, D.; Peral, J. A Computational Method for Enabling Teaching-Learning Process in Huge Online Courses and Communities. *Int. Rev. Res. Open Distrib. Learn.* **2017**, *18*, 225–246.
5. Onah, D.F.; Sinclair, J.; Boyatt, R.; Foss, J. Massive Open Online Courses: Learner Participation. In Proceedings of the 7th International Conference of Education, Research and Innovation (iCERi), Seville, Spain, 17–19 November 2014; pp. 2348–2356.
6. Al-Abri, A.; Jamoussi, Y.; Kraiem, N.; Al-Khanjari, Z. Comprehensive classification of collaboration approaches in E-learning. *Telemat. Inform.* **2017**, *34*, 878–893.
7. Germann, U.; Joanis, E.; Larkin, S. Tightly Packed Tries: How to Fit Large Models into Memory, and Make them Load Fast, Too. In Proceedings of the NAACL HLT Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Boulder, CO, USA, 5 June 2009; pp. 31–39.
8. Peral, J.; Maté, A.; Marco, M. Application of Data Mining techniques to identify relevant Key Performance Indicators. *Comput. Stand. Interfaces* **2017**, *50*, 55–64.

9. Hazem, A.; Boussaha, B.E.A.; Hernández, N. MappSent: a Textual Mapping Approach for Question-to-Question Similarity. In Proceedings of the International Conference Recent Advances in Natural Language Processing, Varna, Bulgaia, 2–8 September 2017; pp. 291–300.
10. Nakov, P.; Màrquez, L.; Moschitti, A.; Magdy, W.; Mubarak, H.; Freihat, A.; Glass, J.; Randeree, B. SemEval-2016 task 3: Community Question Answering. In Proceedings of the 10th International Workshop on Semantic Evaluation. Association for Computational Linguistics, San Diego, CA, USA, 16–17 June 2016.
11. Nakov, P.; Hoogeveen, D.; Màrquez, L.; Moschitti, A.; Mubarak, H.; Baldwin, T.; Verspoor, K. SemEval-2017 task 3: Community Question Answering. In Proceedings of the 11th International Workshop on Semantic Evaluation. Association for Computational Linguistics, Vancouver, BC, Canada, 3–4 August 2017.
12. Fredkin, E. Trie Memory. *Commun. ACM* **1960**, *3*, 490–499.
13. Briandais, R. File Searching Using Variable Length Keys. In Proceedings of the AFIPS Western Joint Computer Conference, Los Angeles, CA, USA, 3–5 March 1959; pp. 295–298.
14. Black, P.E. "Trie", in Dictionary of Algorithms and Data Structures. Pieterse, V., Black, P.E., Eds.; 2011. Available online: http://www.nist.gov/dads/HTML/trie.html (accessed on 16 October 2018).
15. Jung, M.; Shishibori, M.; Tanaka, Y.; Aoe, J. A dynamic construction algorithm for the Compact Patricia trie using the hierarchical structure. *Inform. Process. Manag.* **2002**, *38*, 221–236.
16. Black, P.E. Directed Acyclic Word Graph, in Dictionary of Algorithms and Data Structures. Pieterse, V., Black, P.E., Eds.; 2011. Available online: http://www.nist.gov/dads/HTML/directedAcyclic WordGraph.html (accessed on 16 October 2018).
17. Blumer, A.; Blumer, J.; Haussler, D.; McConnell, R.; Ehrenfeucht, A. Complete inverted files for efficient text retrieval and analysis. *J. Assoc. Comput. Mach.* **1987**, *34*, 578–595.
18. Sánchez-Martínez, F.; Carrasco, R.C.; Martínez-Prieto, M.A.; Adiego, J. Generalized Biwords for Bitext Compression and Translation Spotting. *J. Artif. Intell. Res.* **2012**, *43*, 389–418.
19. Adiego, J.; Brisaboa, N.R.; Martínez-Prieto, M.A.; Sánchez-Martínez, F. A two-level structure for compressing aligned bitexts. In Proceedings of the 16th String Processing and Information Retrieval Symposium, Saariselkä, Finland, 25–27 August 2009, Volume 5721, pp. 114–121.
20. Chang, M.; Poon, C.K. Efficient phrase querying with common phrase index. *Inf. Process. Manag.* **2008**, *44*, 756–769.
21. Kanda, S.; Morita, K.; Fuketa, M. Practical String Dictionary Compression Using String Dictionary Encoding. In Proceedings of the International Conference on Big Data Innovations and Applications, Prague, Czech Republic, 21–23 August 2017.