International Journal of Web Engineering 2015, 4(1): 1-14
DOI: 10.5923/j.web.20150401.01

The Internet Search Engines Based on Artificial Neural Systems Implemented in Hardware would Enable a Powerful and Flexible Context-Based Research of Professional and Scientific Documents

Luca Marchese

Genova, Italy

Abstract This document describes a feasibility study on the development of a new generation of Internet Search Engines that should enable the professional user to search documents by complex contexts instead of single words linked by logical conditions. The method is well known in the literature and is, often, correlated to the WEB-SOM neural methodology. This paper approaches the problem with a new perspective that uses a hierarchical structure of the contexts and the latest neuromorphic VLSI chip technology. The user can search documents giving as input a reference document that is used by the neural search engine in order to find similar documents.The methodologies analysed in this paper share the same neuromorphic hardware technology and the responsiveness to the document context. However different architectural approaches are described. One of the proposed methods is based on a user-friendly X3D (successor of the Virtual Reality Modeling Language) GUI (Graphical User Interface). The other proposed methods use simpler GUIs that seem to be less attractive but, perhaps, more effective.

Keywords Self Organizing Map, WEB-SOM, Internet Search Engine, Radial Basis Function, Neural VLSI, X3D, Context-Based Data Base Navigation

1. Introduction

This paper describes a feasibility study of an Internet Search Engine based on context instead of a collection of single words. The main difference is that the importance of any single word builds the context. If the word is repeated many times in a document, such a word assumes a higher importance. The vector composed of the number of repetitions of meaningful words constitutes the "fingerprint" of the document. The standard way, to search a document, is by typing some words that the desired document must contain. The context-search modality is by writing a small prototype of the document we are searching. In such a way, the input could be a document and the output is a list of documents strongly correlated with it.

There have been many studies on this argument, and they were prevalently based on the SOM (Self Organizing Map) developed by Teuvo Kohonen [1-4]. These studies are, commonly, known as WEB-SOM.

* Corresponding author:
luca.marchese@synaptics.org (Luca Marchese)
Published online at http://journal.sapub.org/web

The architecture of the proposed search engine is based on the RBF (Radial Basis Function) neural network [5]. More precisely, it is based on the RBF hardware implementation in the commercial neural chip CM1K [6-8].

1.1. The WEB-SOM Model

WEB-SOM means WEB Self-Organizing Maps and is an algorithm that orders an information space [1-4]. The map places similar documents in closed spaces. The order helps in finding related documents once any interesting document is found. Fig.1 shows the basic principles of the method and Fig.2 shows a typical map obtained with such algorithm. The software programs, designed with this methodology, typically produce bi-dimensional maps containing "agglomerates" of similar documents. Moving the cursor on such agglomerates the most used words are displayed to the user in order to inform about the context of these documents. There are many different implementations of the WEB-SOM methodology that produce various types of graphical outputs and are, sometimes, organized in hierarchical structures. The main characteristic of the WEB-SOM methodology is the self-organization that characterizes the SOM neural network, often cited as Kohonen Map.

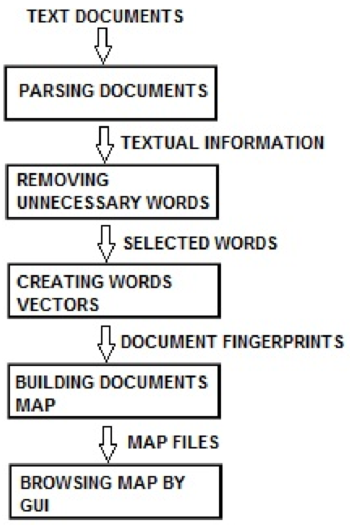


Figure 1. The WEB-SOM methodology

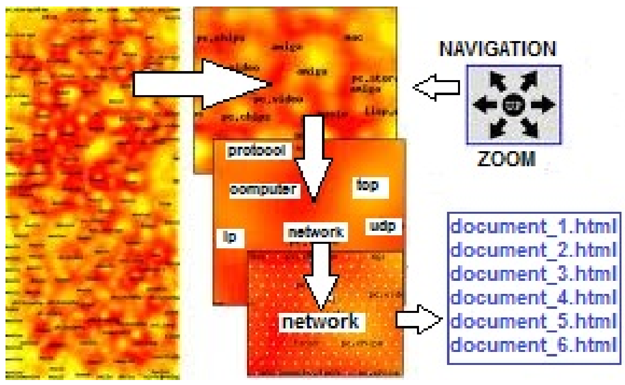


Figure 2. The typical output map of the WEB-SOM. Some words identify the concentration of documents. A dark color indicates a more populated area. If the user moves the cursor to the words, the browser shows a list of documents in another window

2. The Proposed Methodology

The methodology proposed in this paper is different from the WEB-SOM approach in many aspects. The first is that it uses a supervised neural network model (Radial Basis Function). The second difference is that the hierarchical structure of clusters organizes different levels of sensitivity to the fingerprint of the documents. Another important difference is the use of the latest commercially available neuromorphic VLSI technology for pattern recognition. Such a technology should enable the servers of the Internet Search Engine to manage multiple concurrent queries of fingerprint comparisons with the database. Furthermore, I propose an X3D (successor of VRML) browser that enables the user to navigate in a 3D world filled by a hierarchical structure of clusters of documents. The experience is that of moving in a bookstore with nested rooms. As we add new words, we contribute to specialize the desired context and, automatically, move in a room, or we enter in a more specialized sub-room. Direct moving commands could help the navigation. If the user has a reference document, in any format, he can upload directly such a document with the browser. The complex fingerprint extracted by the Internet Search Engine will be compared with the entire database. The Internet Search Engine will respond with a list of similar documents.

This technology has been already presented, in 2000, by L.Marchese at fourth "International Conference on Cognitive and Neural Systems" [9], and at "First International Zero Instruction Sensory Computing Conference" [10]. The concept of "Pattern Recognition Neural Server" was already presented by L.Marchese in 1999 at third "International Conference on Cognitive and Neural Systems" [11] and ANNIE 2000 [12] [13]. In the year 2000 the demand and requirements of the Internet users were not sophisticated as today, and this proposal arrived, probably, to early. Furthermore, the commercial silicon neuromorphic technology was, at that time, limited to 36 neurons in a chip [14] while the current technology has 1024 neurons in a chip [6-8].

2.1. The Radial Basis Function Model and the CM1K Neuromorphic Chip

A radial basis function network (Fig.3) is a neural network that uses radial basis functions as activation functions [5]. It is capable of representing complex nonlinear mappings and is widely used for function approximation, time series prediction and control.

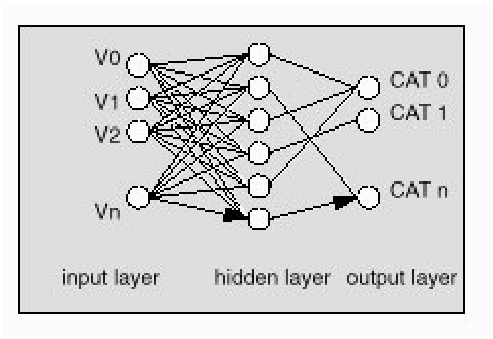


Figure 3. Radial Basis Function architecture

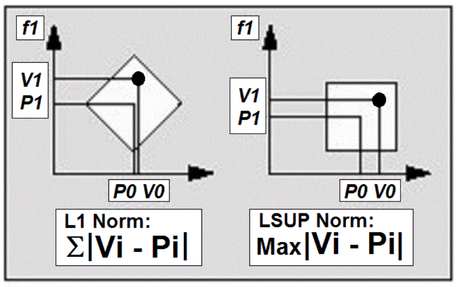


Figure 4. L1 and LSUP distance calculation

The CM1K chip is a hardware implementation of an RBF neural network. The neurons are capable of ranking similarities (L1 or LSUP distance as described in Fig.4) between input vectors and the reference patterns they hold in memory [15]. The neurons report conflicting responses or cases of uncertainty, as well as unknown responses or cases of anomaly or novelty. The time necessary to obtain a response is independent of the number of committed neurons in the network.

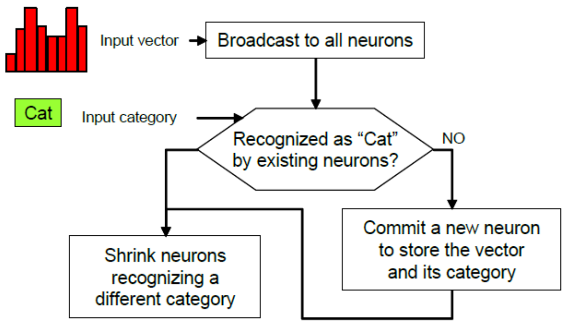

Input vector
Broadcast to all neurons
Cat
Input category
Recognized as "Cat" by existing neurons?
NO
Commit a new neuron to store the vector and its category
Shrink neurons recognizing a different category

Figure 5. Learning process flow chart in CM1K

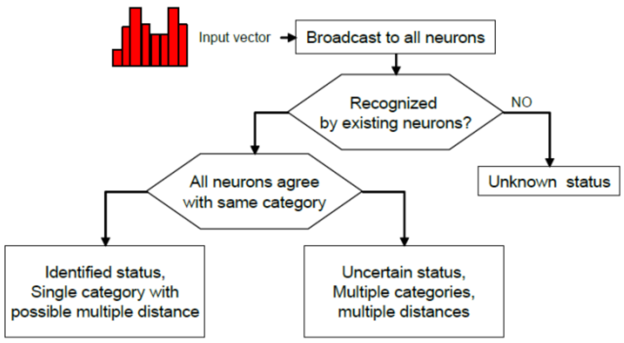

Input vector
Broadcast to all neurons
Recognized by existing neurons?
NO
Unknown status
All neurons agree with same category
Identified status, Single category with possible multiple distance
Uncertain status, Multiple categories, multiple distances

Figure 6. Recognition process flow chart in CM1K

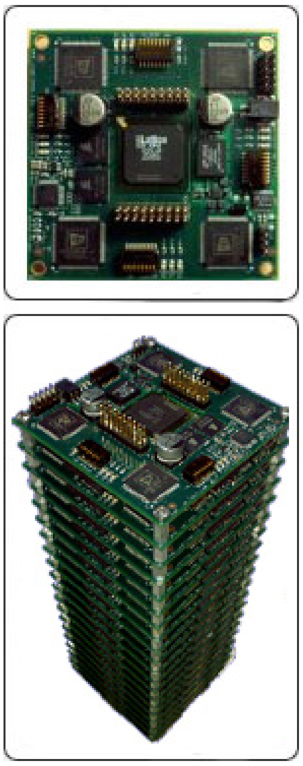


Figure 7. A CM1K stackable board (NeuroStack) and a stack of multiple boards in daisy-chain connection. A stack ofmultipleboards works like a large neural network

The model generator built-in the CM1K chip makes it possible to learn examples in real-time when they drift from the knowledge residing in the current neurons. The "novelty" examples can be stored in neurons assigned to a different context to allow a supervised verification and to learn at a later time.

The ability to assign the neurons to different contexts or sub-networks allows building hierarchical or parallel decision trees between sub-networks. This behavior leads to advanced machine learning with uncertainty management and hypothesis generation. Fig.5 and Fig.6 show respectively the learning process and the recognition process flows in the CM1K chip. Fig.7 shows the board Neuro Stack that contains four CM1K chips for a total of 4096 neurons and a stack of multiple boards [16]. The user can stack these boards together, and they self connect in daisy-chain: two boards constitute an RBF neural network of 8192 neurons and so on.

2.2. Dictionary Building Process

Many WEB-SOM papers describe this process that has the purpose to build a dictionary of meaningful words by analyzing a very large collection of documents. The meaningful words are those that can be considered useful to discriminate between contexts. In this paper, I briefly analyze this process introducing some new algorithmic proposals. I must point out that the scope of the mentioned studies on the WEB-SOM was quite limited. I think that other methodologies could be more efficiently applied to build a dictionary of meaningful words in a comprehensive context like that of the entire Internet contents. In this study, I use the CM1K in order to create the dictionary containing all the meaningful words for arguments discrimination. However, I want suggest that a hash algorithm could efficiently perform this procedure on a conventional computer. The CM1K can compare the current word with all the stored words in parallel mode without the use of hash algorithms and then increase the counter associated with the firing neuron category. In order to perform this operation, we must set MIF=0 and MAF=0 (Minimum Influence Field and Maximum Influence Field [15]). Indeed we need a perfect match between the pattern and the prototype: words must be equal and not similar. Therefore, the L1 or LSUP distance must be null. A same delimiter character must be appended at the end of each word to allow differentiating words with the same root. Example: "house" and "houses". If "houses" has been learned and you broadcast "house", it will be recognized as an exact match. If "houses/" has been learned and you broadcast "house/", they will be considered different. Two counters are associated with any neuron. The first counts the number of times the word is present in all documents (TCNT). The second counts the number of documents in which the word has been found (DCNT). Finally, elaborating this information, we can decide which words are meaningful in order to discriminate between contexts. A conventional computer program implements these counters that are external to the CM1K. Fig.8 shows the procedure flow of this process. When we have stored the words on the CM1K(s) memory, we need to optimize the dictionary by removing words not meaningful for context discrimination. I propose two rules in order to perform this operation:

Meaningful_Flag [Word [Class]] = True

if both the following conditions are true:

1. (Min_1< TCNT [Class] < Max_1) = True
2. (Min_2< DCNT [Class] < Max_2) = True

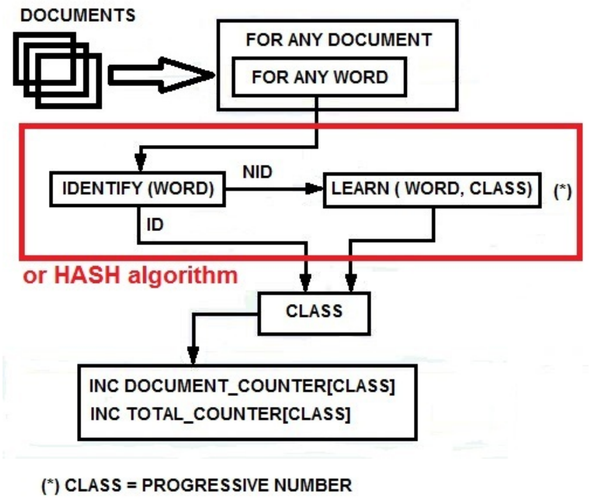


Figure 8. The dictionary building process with CM1K or hash algorithm

If (TCNT < Min_1) then the word is used too few times to be associated with a context.

If (TCNT > Max_1), then word is probabilistically evaluated as a common use word. Therefore, it is useless for context discrimination. Min_1 and Max_1 should be empirically or statistically evaluated. Min_2 = (tot_no_of_analyzed_docs/ G) * (W/100). The context granularity of the database is forced to G (the maximum number of contexts/categories). If the number of documents containing the specified word is lower than W% of the mean dimension of one context/category, then the word is not valid for the context/category discrimination. Max_2 = (tot_no_of_analyzed_docs * C) / G. If the number of documents containing at least one element of the specified word is matching more than C contexts/categories, the word is considered not valid for the context/category discrimination. TCNT is the total counter of words while DCNT is the counter of documents containing the word.

When the Boolean vector of meaningful words is complete, it is possible to optimize the memory of the CM1K(s) by removing the not meaningful prototypes (those prototypes associated with a false flag). The above mentioned operation can be easily done by switching the CM1K chip or chain of chips, to a Save/Restore mode. The synaptic memories and their associated CLASS can then be accessed sequentially for the purpose of reading their content. For each neuron indexed by NEURON_INDEX, its content can be read and saved to file if the Boolean vector [NEURON_INDEX] is false. The resulting file contains only the meaningful words. The CM1K can be cleared and reloaded with the file (which is also portable to other systems).

The number of neurons utilized (equal to the number of the valid words) is the dimension of the "fingerprint" of documents.

Documents clustered by source and clusters logically ordered will result in better context linearity in the dictionary. This property can be an important improvement in the behavior of the document vector compression.

2.3. Documents Fingerprint Extraction and Compression

We build a histogram of any dictionary word to extract the document fingerprint. The document fingerprint is a vector with the dimension of the dictionary. This process will be quickly performed by reading any word of the document, broadcasting it to the CM1K and identifying if a neuron fires. This information is available in the Network Status Register (NSR) of the chip and can be read in one clock cycle after the last character of the word has been submitted. Fig.9 shows this procedure. Alternatively we could use a Hash algorithm because the identification of the word is, again a “perfect match” comparison.

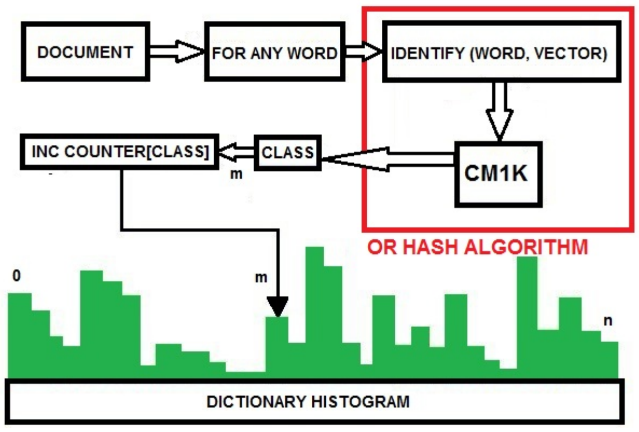


Figure 9. The document fingerprint extraction process

We have an N-dimensional vector for any document of the collection. N is the dimension of the dictionary and is too large for any successive identification (N > 10,000). We require a compression of the vector that does not affect the similitude between couples of vectors. More precisely, the compression must not affect the Euclidean distances between any possible couple of vectors (in our case the Euclidean distance is replaced by the L1or LSUP).

In the WEB-SOM work (Kohonen), a compression based on vertically normally distributed random bits matrix was used in order to compress the dictionary vector (Table.1). The formula used to compute the new reduced vector is:

$$V'[n] = \sum_0^M V[m] \times Bit[n][m] \qquad (1)$$

Table 1. Random Bits Matrix for compression M to N. The component <n> is computed as the sum of the components of the original vector corresponding to bits=1. The number of bits=1 in each row is fixed

	IN 0	IN 1	IN 2	IN 3	IN 4	IN 5	IN 6	IN 7	IN 8	…	IN M
OUT 0	0	1	0	0	0	1	0	0	1	…	0
OUT 1	1	0	0	0	0	1	1	0	0	…	0
OUT 2	0	0	1	0	1	0	0	1	0	…	0
OUT 3	1	0	0	0	1	0	0	0	1	…	0
…	…	…	…	…	…	…	…	…	…	…	…
OUT N	0	1	0	1	0	0	0	0	0	…	1

Table 2. Random Bits Matrix for compression M to 256. The components of the original vector of size M are selected if the corresponding bit value is 1. There are 256 bits=1 in each row. The table builds N vectors of 256 components. It must be N=255 in order to use CM1K in the following processes.

	IN 0	IN 1	IN 2	IN 3	IN 4	IN 5	IN 6	IN 7	IN 8	…	IN M
VECT 0	0	1	0	0	0	1	0	0	1	…	0
VECT 1	1	0	0	0	0	1	1	0	0	…	0
VECT 2	0	0	1	0	1	0	0	1	0	…	0
VECT 3	1	0	0	0	1	0	0	0	1	…	0
…	…	…	…	…	…	…	…	…	…	…	…
VECT N	0	1	0	1	0	0	0	0	0	…	1

We use the CM1K to perform the compression task. We have vectors with 10,000 components and need to have equivalent vectors with 256 components. The equivalence is referred to the normalized distance between couples of vectors: we do not use the Euclidean distance but the L1 or LSUP distance in order to have compatibility with the CM1K. The final vector should be recognizable by the CM1K (256 components byte wide comparison) in one single recognition operation. The compression is performed using a matrix with 256 randomly distributed bits "1" in each row that builds a 256 components vector as a result of the AND operation with the original vector (Table 2). The CM1K(s), previously trained with fixed random patterns recognize the vector, and the matching class is one component of the new compressed vector. The training set is composed of 256 patterns (with 256 components) randomly generated. These patterns are sorted on the basis of the L1 distance with a K-Nearest-Neighborhood procedure and assigned to an incremental category number. Furthermore, the reference patterns must have a minimal distance (a defined threshold) from the nearest patterns assigned to the lower and higher category number (Fig. 10b). We repeat this step with different Boolean vectors for any component of the new compressed vector (Fig.10a). This compression is more reliable than the simple Bits Matrix compression. The new vectors conserve the similitude property better than the random bits matrix processed vectors (Fig.11 and Fig.12). This property is true if the documents contain a quite high number of words of the dictionary. The system behaves better working on the database of large documents, as well as any other text-analysis algorithm. The test to verify the validity of the method is explained below.

We build a database of pseudo-random vectors composed of 10,000 components and choose a reference vector. The reference vector is compared with all the other vectors by using a normal program of L1 distance computation. We insert in the list all the vectors having normalized distance < D_MAX (2). This process is very long and could be performed quickly using an appropriate algorithm on the CM1K. However, we can use a simple program on a conventional computer because it is needed only one time for a test.

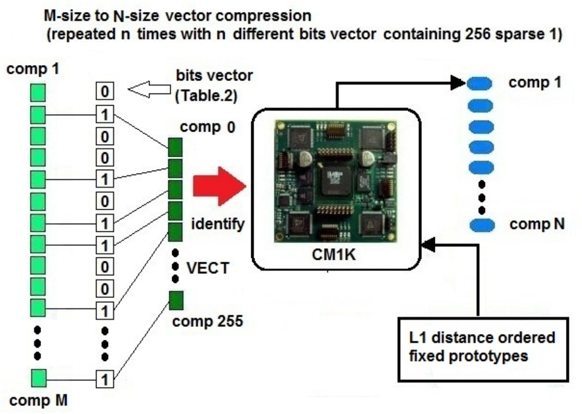

M-size to N-size vector compression
(repeated n times with n different bits vector containing 256 sparse 1)
comp 1
bits vector (Table.2)
comp 0
identify
VECT
comp 255
comp M
CM1K
comp 1
comp N
L1 distance ordered fixed prototypes

Figure 10a. Random-bits-matrix compression with L1 or LSUP distance recognition based on random reference prototypes. The reference prototypes are ordered by L1 distance (K-nearest-Neighborhood) and assigned to an incremental category number (Fig.10b). N=256.

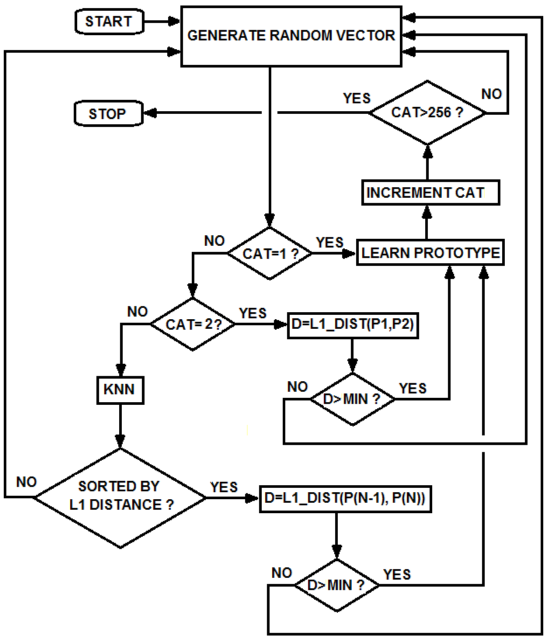


Figure 10b. The flow-chart of the procedure used to sort the reference prototypes by L1 distance. Any randomly generated vector is compared with all the previous vectors, and they are sorted by L1 distance with the K-NN algorithm (K-Nearest-Neighborhood)

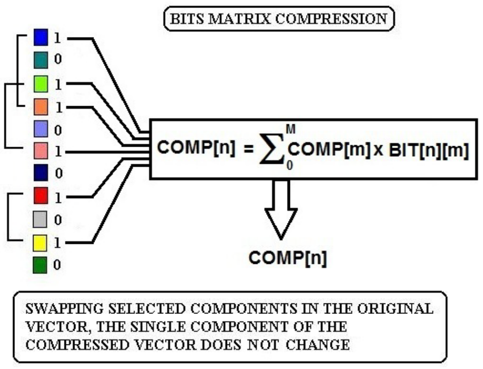

BITS MATRIX COMPRESSION
1
0
1
1
0
1
0
1
0
1
0
COMP[n] = $\sum_{0}^{M}$ COMP[m] x BIT[n][m]
COMP[n]
SWAPPING SELECTED COMPONENTS IN THE ORIGINAL VECTOR, THE SINGLE COMPONENT OF THE COMPRESSED VECTOR DOES NOT CHANGE

Figure 11. The standard random-bit-matrix compression methodology: if we swap couples of selected components of the original vector, the single component of the compressed vector does not change

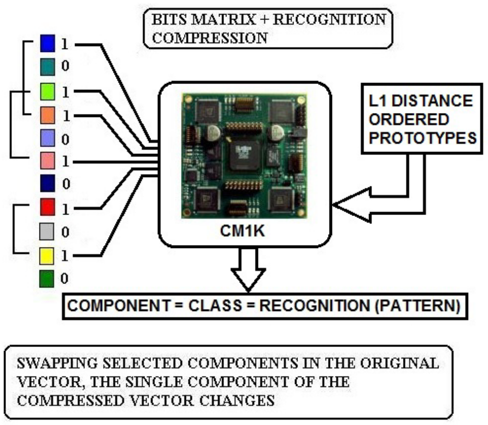

BITS MATRIX + RECOGNITION COMPRESSION
1
0
1
1
0
1
0
1
0
1
0
CM1K
L1 DISTANCE ORDERED PROTOTYPES
COMPONENT = CLASS = RECOGNITION (PATTERN)
SWAPPING SELECTED COMPONENTS IN THE ORIGINAL VECTOR, THE SINGLE COMPONENT OF THE COMPRESSED VECTOR CHANGES

Figure 12. In the proposed random-bits-matrix compression followed by L1 or LSUP distance recognition: if we swap couples of selected components of the original vector, the single component of the compressed vector changes as required

We repeat the process with the equivalent compressed vectors (256 components). Then we insert in the second list the vectors having normalized distance < D_MAX. The equivalence of the two lists should mean a 100% conservation of the similitude: a value over 70% is acceptable. We repeat the test with many target vectors. The standard deviation (3) of the results obtained should be very limited in order to perform a reliable test. At the end of this process, the average of the results should be considered.

$$\begin{aligned} &D = L1\_d / N \\ &D = normalized\_L1\_dist \\ &L1\_d = L1\_dist \\ &N = vector\_size \end{aligned} \tag{2}$$

$$\begin{aligned} &\sigma = \sqrt{\frac{\sum_{n=1}^{N}(x-\mu)}{N}} \\ &N = number\_of\_samples \\ &x = sample\_result \\ &\mu = mean\_result \\ &\sigma = stnd\_dev \end{aligned} \tag{3}$$

2.3.1. The Output of This Process

The output of this process is a list of records composed of [URL] [VECTOR] [MUW] related to the examined documents. The URL is Uniform Resource Locator for the specified document while VECTOR and MUW are respectively the fingerprint and the most used word of the document (Table.3).

Table 3. Association of URL, fingerprint, and Most Used Word

URL	VECTOR	MUW
http://domainX.dir1.docZ.html	[230][234][200]...[010]	transistor
http://domainY.dir1.docW.html	[240][134][002]...[030]	optical
http://domainZ.dir1.docY.html	[130][034][005]...[050]	car
http://domainZ.dir2.docD.html	[030][114][135]...[120]	network
....	....	....

2.4. Hierarchical Clustering and 3D Mapping

As I have above explained, the system is based on a hierarchical structure of clusters or "containers" that we can see as "nested" rooms of a bookstore. The level <0> is the most detailed level or, using the bookstore simile, is the room where we can find, finally, the books we are searching. The hierarchy has a specific depth. The upper levels are here, generically, indicated as level <n>.

2.4.1. Level<0>Clustering

When we have a table with records <URL> <VECTOR> <MUW> we must put them into clusters basing this operation on the distances between their vectors that I have called "fingerprints."

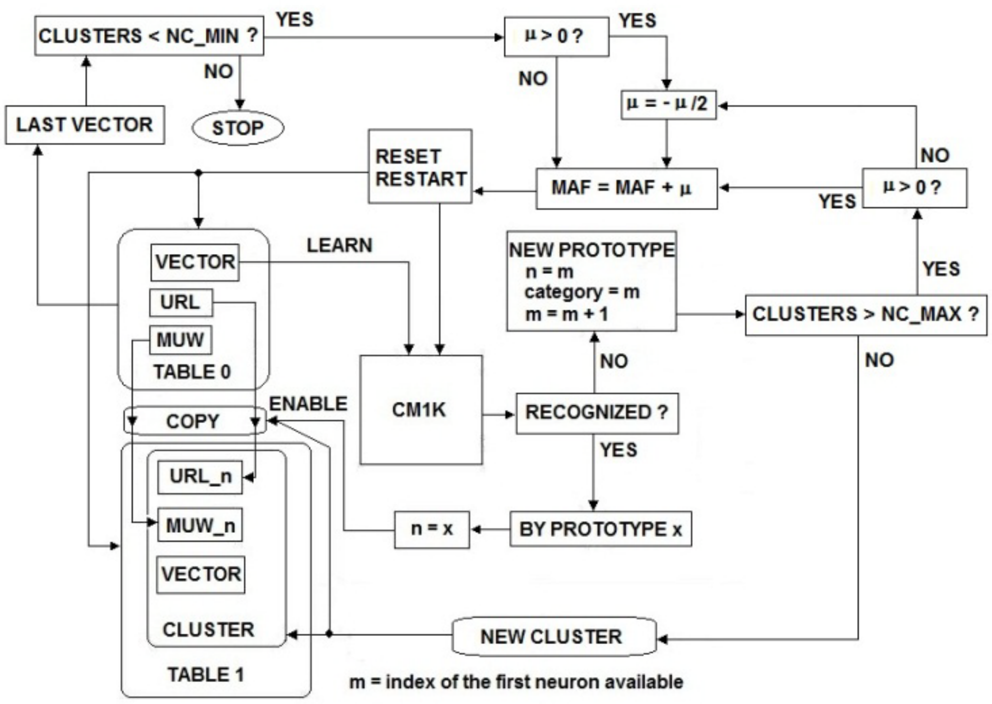


Figure 13. Clustering process from documents to level <0> clusters. There is a loop regulated by a dichotomic algorithm that adjusts the MAF parameter of the CM1K neural-processor in order to reach the desired number of clusters (between NC_MIN and NC_MAX)

Table 4. Example of records for the key CLUSTER = 23403 at the level <0>. There is an associated vector or fingerprint of the cluster. The cluster contains a collection of URLs (documents). Each URL has an associated Most Used Word

CLUSTER_0	VECTOR	URL	MUW
23403	[120] [030] ... [240]	http://domainX.dirX.docX.html	neural
		...	...
		http://domainX.dirX.docX.html	Kohonen

Table 5. The table contains many clusters at the level<0>. A cluster contains many URLs and the associated MUWs. The table is composed of many records of Table 4

CLUSTER_0	VECTOR	URL	MUW
…	…	…	…
23403	[120] [030] ... [240]	http://domainX/dirX/docX.html	neural
		...	...
		http://domainX.dirX.docX.html	Kohonen
…	…	…	…
30000	[220] [130] ... [140]	http://domainX.dirX.docX.html	RBF
		…	…
		http://domainX.dirX.docX.html	KNN
…	…	…	…

We want perform this process as fast as possible using the recognition of the Radial Basis Function hardware implementation. The choice of the CM1K neural chip is related to the easy to understand behavior and programming interface and mainly for its expandability at no cost of performance.

The process of clustering (Fig.13) is a hybrid situation where some aspects of supervised and unsupervised learning behave together. We must perform the learning process, without knowing a priori the classes associated with the patterns. At the same time, we need to associate a class with any cluster. The choice is to associate an incremental number linked with the commitment of a new prototype neuron in the Radial Basis Function neural network. Fig13 describes a clustering process which is implemented in a loop where the MAF is progressively adjusted with a dichotomic algorithm until it reaches a user-specified number of clusters. If we consider an Internet Search Engine, DB-Size is around 67 billion pages (2014 estimation of Google indexed pages [17]). As the DB size is typically very large, we should find an optimal start value in order to reduce the number of required cycles. We select an optimal starting value of MAF (Maximum Influence Field) [15] following the relation: MAF = f (DB_Size, NN_Max_Size). DB_Size is the size of the database and NN_Max_Size is the maximum number of neurons available for this clustering level. This relation could be extrapolated from the results obtained with the MAF optimizing loop on smaller databases. When the input pattern does not match the influence field of any other existing prototype, the learning process commits a new neuron. This new neuron has MAF as influence field. The new prototype connects with a new cluster.

The learning process never reduces the influence field of a prototype: the category-mismatch [15] condition cannot happen because any new category is a sequential number.

When the CM1K learns a pattern, we memorize the URL (Uniform Resource Locator) of the document and its most used word (MUW) in a database. The key of such a database is simply the number of the cluster. The relation is "one to many" because any cluster contains many URLs with the associated MUW (Table 4 and Table 5).

2.4.2. Level <n> Clustering

The upper clustering levels supply a hierarchical structure for the navigation of the database. The number of clustering levels is a function of the database dimension and the parameters used to define an effective navigation. Starting from n=1, any cluster of Level<n> does not contain URLs but the Level<(n-1)> clusters. An MUWs list connects with any Level<(n-1)> cluster, and a number identifies any cluster (Table.6 and Table.7). In order to enable a 3D navigation, we need to add to this record some information that represents the x-y-z position of the cluster in 3D space (Fig.14). We perform the recognition of the vector divided into three elements, using CM1K(s) trained with predefined pseudo-random patterns (Fig.15):

X = CLASS (k = 0 - m STEP 3) { V[k] }
Y = CLASS (k = 1 - m STEP 3) { V[k] }
Z = CLASS (k = 2 - m STEP 3) { V[k] }

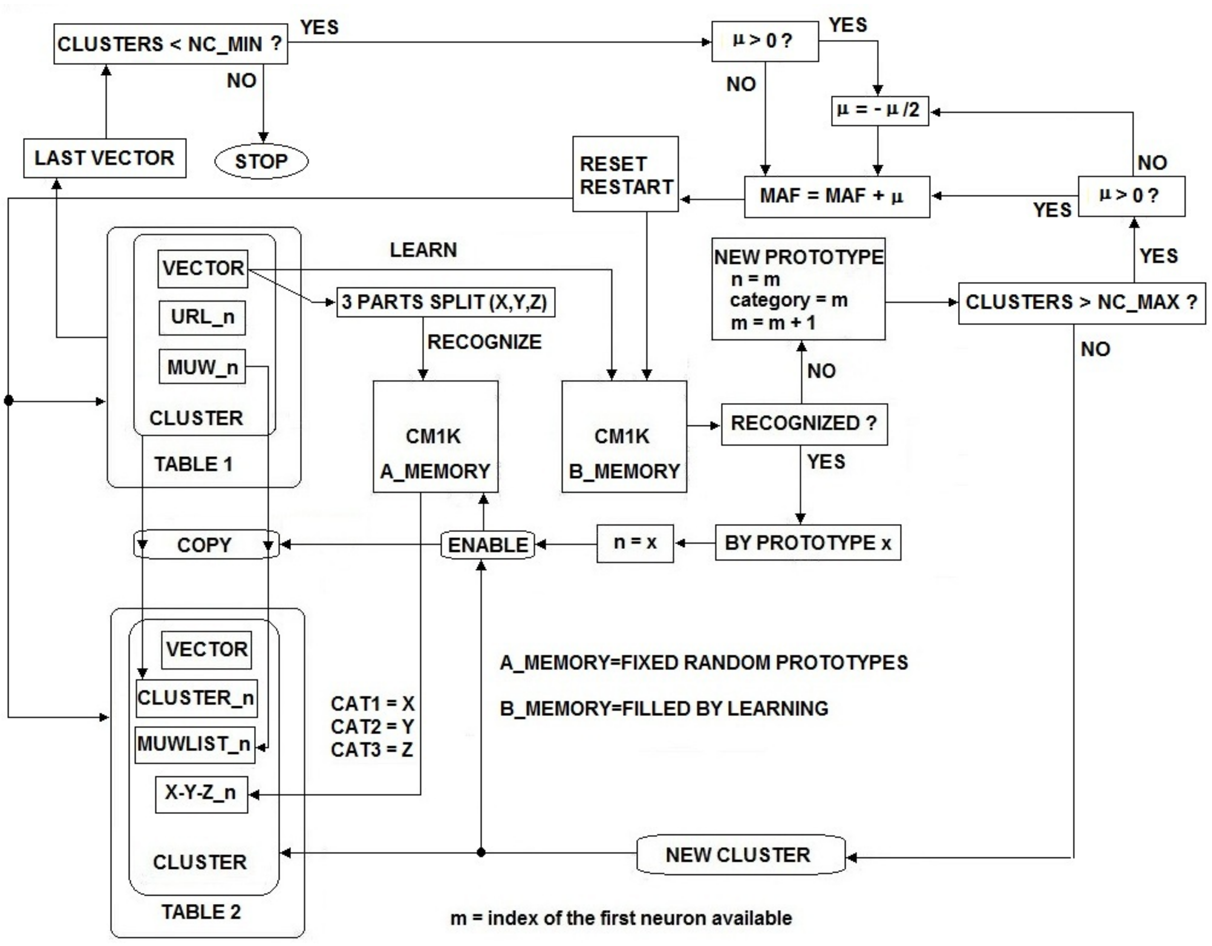


Figure 14. Clustering process from level <n-1> to level <n> cluster. These clustering levels contain the management of 3D navigation

Table 6. Records for the key CLUSTER = 2366 at the level<n>

CLUSTER N	VECTOR	CLUSTER n-1	MUW-LIST	X	Y	Z
2366	[120] [030] … [240]	23403	WORD1, ...n	2404	1230	240
		...	WORD1,...n	...	...	...
		32240	WORD1,...n	2400	1040	200

Table 7. The table with more clusters at level <n>. The sub-clusters of level <n-1> have their MUW-LISTs and coordinates associated

CLUSTER N	VECTOR	CLUSTER n-1	MUW-LIST	X	Y	Z
…	…	…	…	…	…	…
2366	[120] [030] … [240]	1	WORD1, ...n	2404	1230	240
		...	WORD1,...n	...	...	...
		32240	WORD1,...n	2400	1040	200
…	…	…	…	…	…	…
3000	[230] [200] … [100]	1	WORD1, ...n	1600	200	500
		…	WORD1, ...n	…	…	…
		2000	WORD1, ...n	340	2340	1300
…	…	…	…	…	…	…

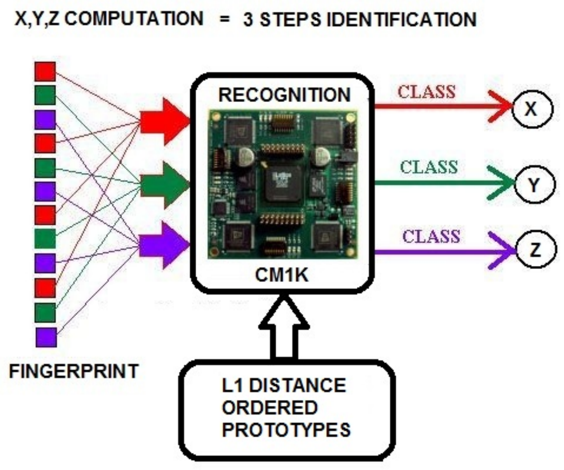

X,Y,Z COMPUTATION = 3 STEPS IDENTIFICATION
RECOGNITION
CLASS
X
CLASS
Y
CLASS
Z
CM1K
FINGERPRINT
L1 DISTANCE ORDERED PROTOTYPES

Figure 15. The computation process of the coordinates. We split the fingerprint into three parts, and the CM1K recognizes these parts separately. We use a database of fixed random patterns as neural memory of the CM1K. We use the class (range 0-16000) as acoordinate

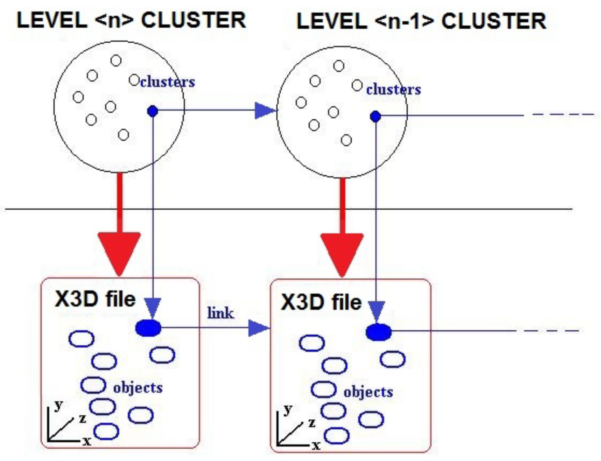


Figure 16. The cluster <n> is associated with anX3D file and contains the clusters <n-1> (objects linked with other X3D files). The server dynamically builds the X3D file from the table. The x, y, z are the coordinates of the objects. The MUW-LIST associated to the object is displayed on the browser when the mouse pointer is on the object

2.5. From Clusters to X3D

The process described here is required to obtain a full X3D hierarchical description of the database space.

X3D (successor of VRML) is featured to describe three-dimensional spaces and environments containing three-dimensional objects. X3D files can be interpreted by X3D browsers. An X3D browser enables the navigation of the spaces described by X3D files. Our objects are very simple geometrical shapes because they must represent the clusters as “containers” or “rooms”.

Fig.16 shows how the data contained in cluster tables are used to build X3D files.

Fig.14 shows the process that creates a hierarchical structure of X3D files starting from cluster tables.

2.6. X3D Data Base Navigation

The first access to the database is the highest level X3D file in the hierarchical structure. By moving in this space, we can see many objects representing the lower level clusters positioned dependently of their associated vector.

Therefore, we can see zones more populated of objects than others.

We can walk, fly, translate, tilt, yaw, pitch and roll using keys on the browser or a specialized joystick. When the mouse pointer is on an object, the browser shows, in another frame, the list of words associated with the cluster that is represented by this object (Fig.17). The X3D file contains this list of words. Looking at these words, we can understand the contents of the cluster and decide if it is interesting for our research. In this case,we can click on the object obtaining the download and visualization of the associated X3D file.

This type of navigation looks more enjoyable than useful and must be enhanced to enable us to move in this world by inputting complex context information.

2.7. Context-Based Navigation

We need to move in the 3D space not only by making direct X3D movements, but also by putting context information. There are two possible scenarios of this behavior. In the first, we have a "reference document" and want to find similar documents. In this case, the clustering processes explained in Figs.13, and 14 can manage the navigation. We can upload the "reference document" with the browser to the Internet Search Engine. The Internet Search Engine extracts the fingerprint and compares it with the database using a Pattern Recognition Neural Server based on CM1K boards (Fig.19). We are then automatically transported to the 3D world in the sub-space addressed by the fingerprint of the "reference document". We could select if we want confirm manually any sub-cluster passage. If the fingerprint of the "reference document" is enough detailed, we could be directly transported to the deepest bookstore room containing the "books". If the fingerprint is not enough detailed or it is not within the "influence field" of any deepest "room", we could be stopped at any clustering level in any x-y-z position. In such a case, we should navigate from that point with direct 3D movements helped by the MUW-list that is shown in the window of the browser when the mouse pointer is on the cluster.

In the second scenario, we do not have a reference document and we need to build incrementally a context by typing words in the window. In this second case, a real fingerprint cannot be constructed reliably because we should be asked to input too many words and indicate the "weights" of them for the context. In order to manage a navigation driven by a small reference context, I have proposed the clustering process of Fig.18. Comparing such a process with that of Fig.14, we can note that I have added the management of a new table "word-to-coordinates". Fig.17 explains the navigation process. When we input a new word, the server calculates the contribution of such a word to build a new position and sends these coordinates to the client browser. The server and the client share the computation of the new position. The server sends the coordinates associated with that word in a specific cluster. The server finds the coordinates in the corresponding table "word-to-coordinates". The client (the browser) averages these coordinates with the current coordinates in order to build the final position. The clustering process of Fig.18 builds a word-to-coordinates table for any cluster at any clustering level. This navigation process does not require the pattern recognition capability of the Neural Server, but it is not exactly the most important target of this feasibility study. I believe that the real revolution in the research of scientific and professional documents is the process that uploads the "reference document" as shown in Fig.19. The browser should accept any document format (HTML, Txt, RTF, Pdf, Word, PostScript), and the server should extract the fingerprint. Then, the server asks for a pattern recognition service to the CM1K neural-server and forwards the results (coordinates or cluster number) to the client (browser). In the next paragraph, I propose a different Internet Search Engine based on multiple dictionaries associated with specific disciplines.

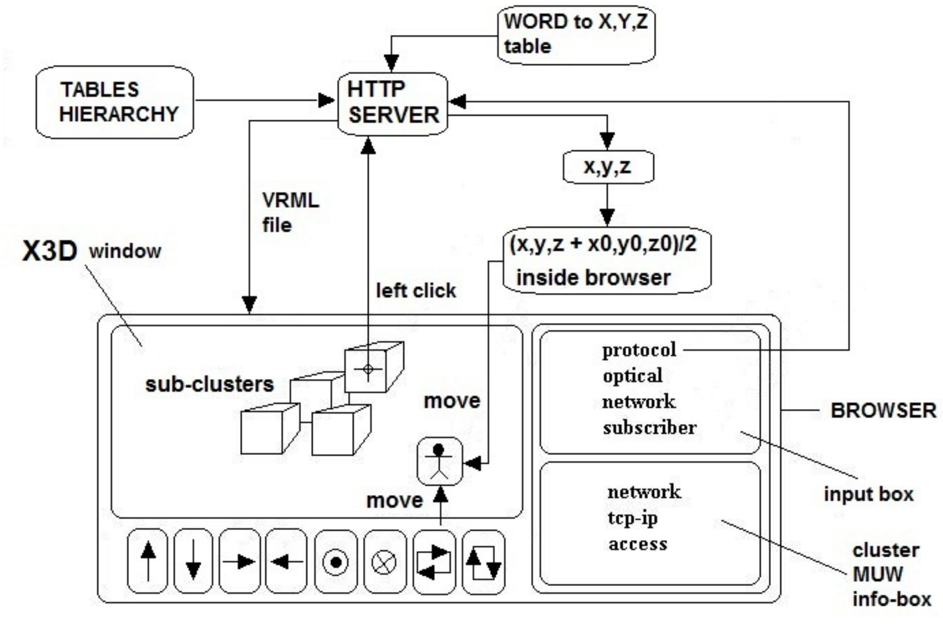


Figure 17. The Internet X3D browser queries the Internet Search Engine with some words that build a small context

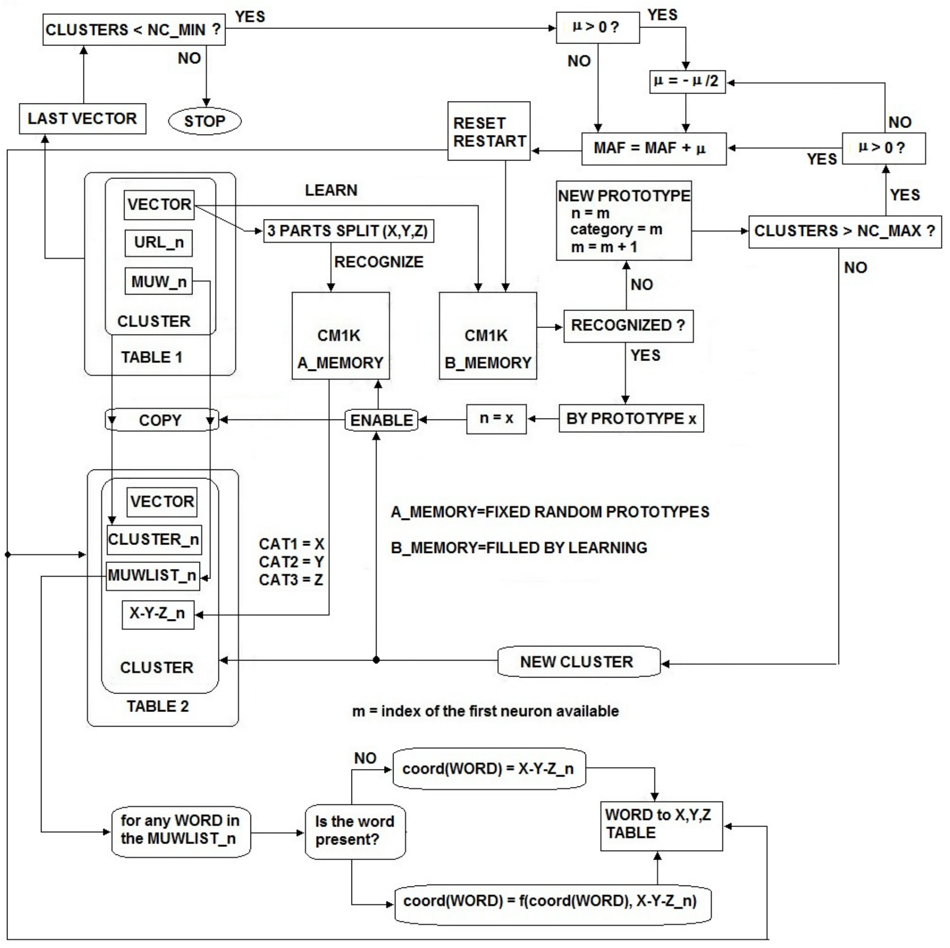


Figure 18. Clustering process from level <n-1> to level <n> with the management of queries containing a small number of words (small context) instead of a real reference document

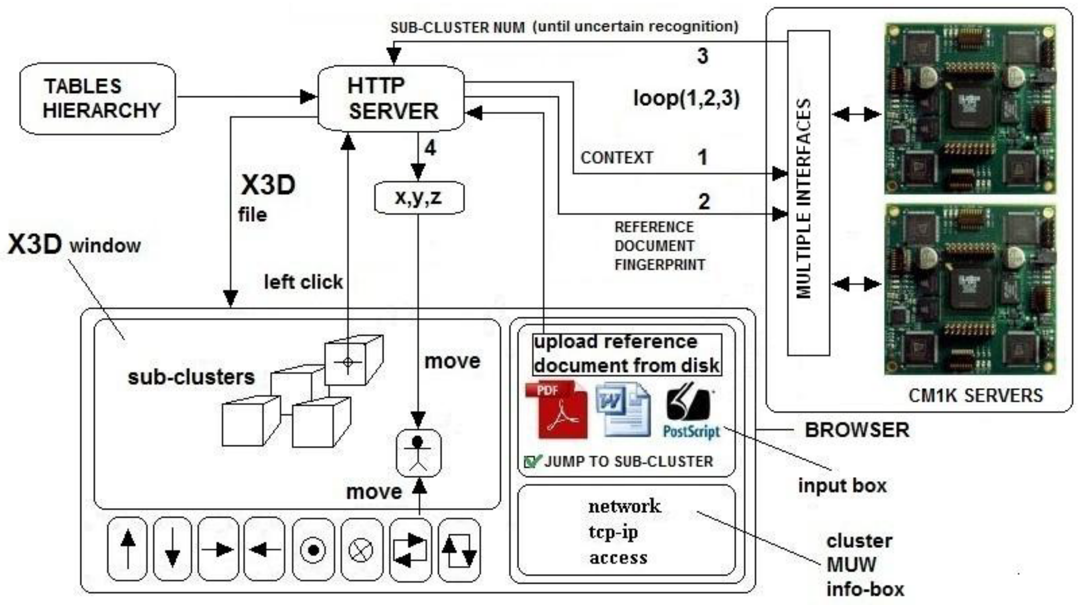


Figure 19. The Internet X3D browser queries the Internet Search Engine with reference documents. The "CONTEXT" sent by the HTTP server to the CM1K neural-server is not referred to the document but is the appropriate synaptic memory for the hierarchy level and the current cluster. The browser enables the user to select the automatic jump inside the next clustering level or perform manually this operation. The host computer connects to multiple independent CM1K servers in order to manage multiple clients requests

3. Fingerprintbased on Specialized Dictionaries

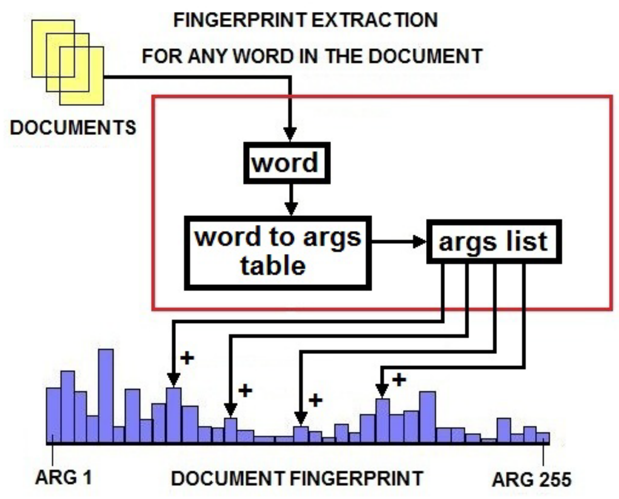


Figure 20. The picture describes the process of fingerprint extraction by using 255 specialized dictionaries

There is an alternative way to build context-based Internet Search Engines. Instead of considering the context in a continuous space we could decide to discretize this space by using a large number of specialized dictionaries dedicated to specific disciplines from economy to science and technology [18][19]. A hierarchy with different levels of specialization organizes the system. Experts in any specific discipline could build specialized dictionaries. A "one to many" database relation should link any specific word to the dictionaries at different levels of the hierarchy. The fingerprint of a document is the number of words linked to any specific dictionary. The fingerprint size is the number of dictionaries for the current level of the hierarchy. The server uses the database of relations "word =>dictionary" in order to build the fingerprint of the document. The Internet Search Engine, as usually, scans the net contents and builds the fingerprints of the documents updating the database based on Table 3, Table 4 and Table 6. In my proposal, I fix the number of dictionaries for any level of the hierarchy to 256 for compatibility with the maximum dimension of the CM1K vector. The fingerprint of a document is a 256 elements vector that can be computed counting the document words belonging to the 256 specialized vocabularies. From the model described in the first part of this document, only the fingerprint extraction modality is different, and we could consider valid any other process described in this paper. The advantage of this approach is the higher reliability of the fingerprint. The drawback is the need to fix, a priori, a comprehensive hierarchical structure of the global knowledge available on the Internet. Indeed the origin of a new discipline would not be tracked by the system. On the contrary, this new discipline would be tracked by the system based on the global dictionary fingerprint, provided that this discipline does not introduce too many neologisms. Fig.20 shows the fingerprint extraction process based on specialized dictionaries.

4. The Simplest, Fastest and Most Scalable Solution Based on This Neuromorphic Hardware

In this paragraph, we analyze the simplest, most performant and most reliable way to build a Document Search Engine based on context search.

In this solution, we avoid using the hierarchical 3D navigation in favor of a more conventional interface. The fingerprint is based on the compressed vector of words or on indexed controlled dictionaries as explained in paragraph 3. The target could be a specialized search engine for specific disciplines (medicine, electronics, informatics etc.). In the case of specialized search engines, the controlled vocabularies should be related to branches of the main discipline. If the main argument is medicine, then the 256 controlled dictionaries could be related to cardiology, neurosurgery, ophthalmology, etc. In this framework, we avoid the hierarchical clustering process described in 2.4. Instead, we perform a single level of clustering based on a specific minimal granularity (it could be the same of Fig.13). If, as example, the database is composed of 100,000 documents, the clustering process could be tuned to build 10,000 clusters. The averaged content of each cluster is 10 documents. We need 10,000 neurons that we can find in only three NeuroStack boards (~12,000 neurons).

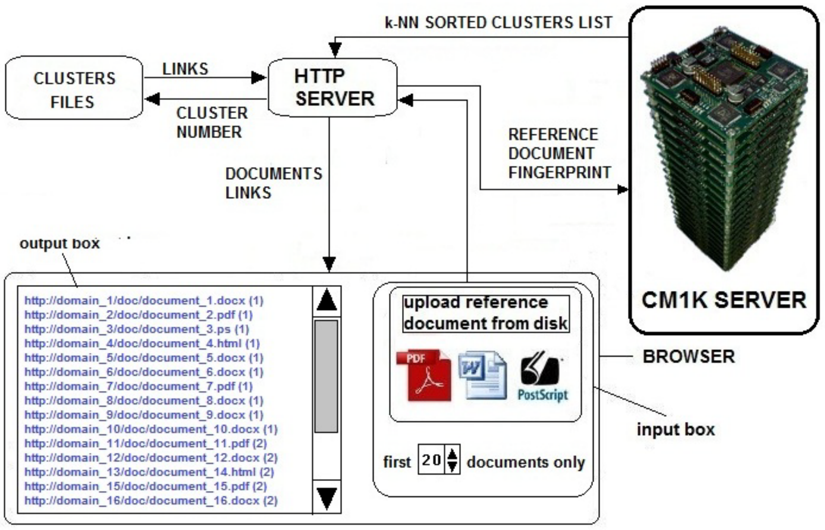
The host processor can query the NeuroStack in k-Nearest-Neighborhood mode, using the fingerprint of the reference document supplied by the client. In k-Nearest-Neighborhood mode, the NeuroStack responds with the entire list of clusters ordered by L1 or L-SUP distance from the input fingerprint. The client can ask for a maximum number of clusters (from 1 to N). The client knows that the links contained in the first cluster are the most similar to the reference document (Fig. 21). Any request from multiple clients is executed in real-time because the reference fingerprint is compared with the 10,000 stored fingerprints in parallel mode. The time required to perform the k-Nearest-Neighborhood sorting process is independent of the number of the stored fingerprints.

5. An Automatic Hybrid Solution

The extreme application would be a Context-Based Internet Search Engine managing the same number of pages that are, currently, indexed by Google (~67 billions of pages estimated in 2014 [17]).

The number of indexed URLs is almost always significantly smaller than the number of crawled URLs. Indeed the total of indexed excludes URLs identified as duplicates or non-canonical, as less useful, or that contain a meta *noindex* tag.

I have presented some possible realizations of context-based research engines with a neuromorphic hardware. However, I want underline that these systems could be synergic and not a replacement of the traditional "key-word" based systems. The user could perform a first research with the traditional system and select a document that satisfies the requirements. Then he can upload the document as reference.



```
http://domain_1/doc/document_1.docx (1)
http://domain_2/doc/document_2.pdf (1)
http://domain_3/doc/document_3.ps (1)
http://domain_4/doc/document_4.html (1)
http://domain_5/doc/document_5.docx (1)
http://domain_6/doc/document_6.docx (1)
http://domain_7/doc/document_7.pdf (1)
http://domain_8/doc/document_8.docx (1)
http://domain_9/doc/document_9.docx (1)
http://domain_10/doc/document_10.docx (1)
http://domain_11/doc/document_11.pdf (2)
http://domain_12/doc/document_12.docx (2)
http://domain_13/doc/document_14.html (2)
http://domain_15/doc/document_15.pdf (2)
http://domain_16/doc/document_16.docx (2)
```

Figure 21. The simplest implementation of the Context-Based Internet Search Engine with neuromorphic VLSI. This configuration can be applied independently of the methodology used for the fingerprint extraction. The system uses only one level of clustering and responds to the client queries with a list of documents sorted with the k-Nearest-Neighborhood algorithm. In this example, the average number of documents contained in a cluster is 10. The client uploads the reference document and asks for the 20 most correlated links. The server responds with the 20 links contained in the first clusters of the k-NN sorted list

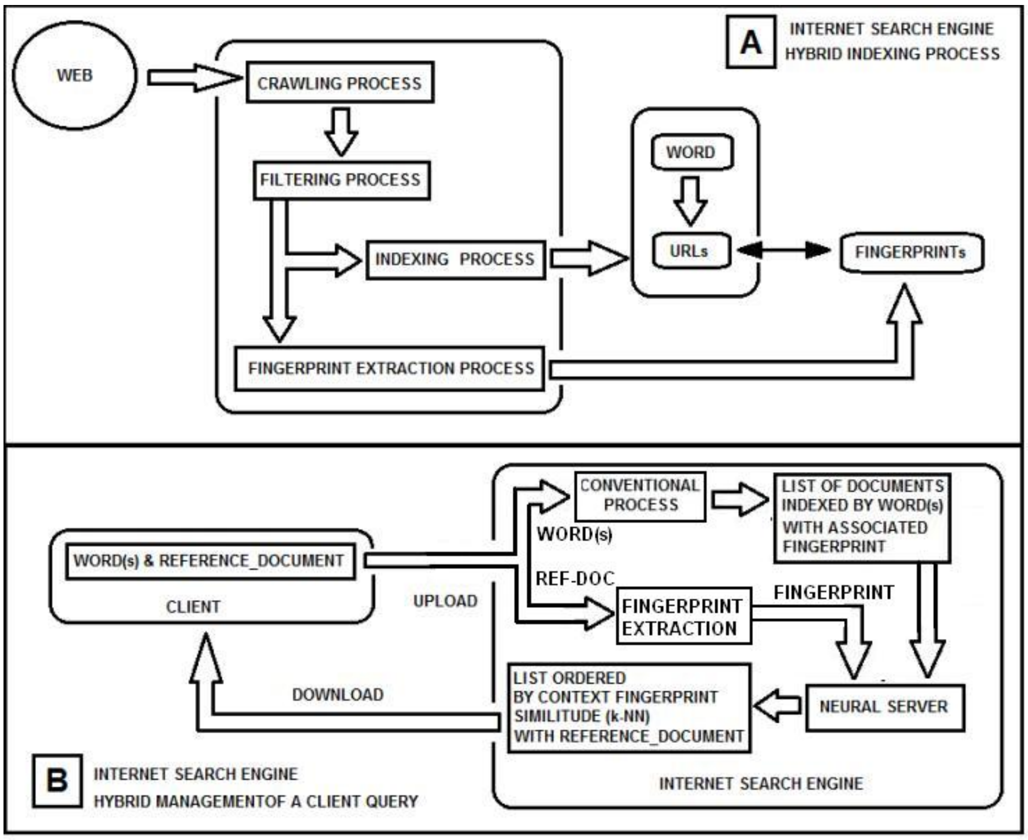


Figure 22. The picture shows the hybrid keywords-context approach. In A the crawling process and the subsequent processes are described. In B, the client sends to the Internet Search Engine the required words (with logic conditions) together with a reference document. The required words generate a first list of documents. The fingerprints of the documents contained in the list are compared with the fingerprint of the reference document, and a sub-list is sent to the client

However, we could consider an automatic hybrid solution where the user sets a logic condition (presence of one or more key-words) together with a reference document. In such a process, the search engine builds a first list of documents using the logic condition and sends this list to the neural server in order to build the final list using the context fingerprint comparison. In order to execute this process, the Internet Search Engine should perform indexing and add a context fingerprint extracted with one of the above-mentioned methods. In this way, the first list of documents contains context fingerprints that can be sent to the neural server (Fig.22).

6. Conclusions

In this paper,I have presented a feasibility study for the realization of a new generation of Internet Search Engines. I have used the latest neuromorphic VLSI pattern recognition technology commercially available and the most sophisticated Internet navigation languages. The pattern recognition specialized chips enable the real-time management of multiple clients queries on a database of document fingerprints. The server must analyze a very large quantity of fingerprints in a fuzzy way that is not suitable for conventional processors. These engines seem more suitable for professional customers that need to search documents by complex contextual information instead of simple collections of words linked by logical conditions. This search system could be synergic and not a replacement of the traditional system. I argue that this type of contextual research will be the best method in the future. We realize everyday that the result of the conventional research by key-words is full of links that are not correlated with our query. The problem is often originated by advertising pages and is well known as "web spam" or "spamdexing" [20]. Google notifies sites that it considers spam at a rate of 40,000-60,000 per month. The mystification of multiple contexts in a page would be very difficult if not impossible.

I believe that this could be the right time for a new generation of Internet Search Engines that search images and documents following a fuzzy "brain-like" philosophy instead of an Aristotelian or Boolean method.

ACKNOWLEDGEMENTS

I want to thank Anne Menendez and Guy Paillet of General-Vision for the technical support on the CM1K chip technology and the feedbacks on this feasibility study. I also want to thank the reviewers for the useful suggestions.

REFERENCES

[1] Kohonen, T. (1995). "Self-Organizing Maps" Springer, Berlin, Heidelberg.

[2] Kohonen, T. (1982). "Self-organized formation of topologically correct feature maps." Biological Cybernetics, 43:59-69.

[3] Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1996). "Exploration of full-text databases with self-organizing maps." Submitted to ICNN-96, Washington D.C.

[4] Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1996). "Creating an order in digital libraries with self-organizing maps" Submitted to WCNN-96, San Diego, California.

[5] Buhmann, Martin D. (2003), "Radial Basis Functions: Theory and Implementations", Cambridge University Press, ISBN 978-0-521-63338-3.

[6] General Vision Inc, "CM1K Technical Manual", www.general-vision.com

[7] General Vision Inc, "CM1K, the simplest API", www.general-vision.com

[8] General Vision Inc, "CM1K The fastest KNN chip", www.general-vision.com

[9] Marchese, L. "Distributed Database content based navigation and Internet search engines powered by neural VLSI". - "Fourth International Conference on Cognitive and Neural Systems"- (27/05/2000) - Department of Cognitive and Neural Systems – Boston University

[10] Marchese, L. "Neural VLSI for Internet and telecommunications technology" - First International Zero Instruction Sensory Computing Conference (LeCorum Montpellier) (27/07/2000).

[11] Marchese, L. "Neuromorphic VLSI server" - "Third International Conference on Cognitive and Neural Systems" (27/05/1999) Department of Cognitive and Neural Systems - Boston University.

[12] Marchese, L. "A neural network chips based server" - ANNIE 2000: Smart Engineering System Design (05/08/2000) (University of Missouri-Rolla).

[13] Marchese, L. "Intelligent engineering systems through Artificial Neural Networks: a neural network chips based server"-ASME PRESS - Editors:Cihan Dagli, Anna L.Buczak, Joydeep Ghosh, Mark J.Embrechts, Ocan Ersoy, Stephen Kercel (2000).

[14] IBM, Silicon Recognition, "ZISC036 Operational Manual".

[15] General Vision Inc, "Neuro Mem Decision Space Mapping Manual", www.general-vision.com

[16] General Vision Inc, "Neuro Mem Technology Reference Guide", www.general-vision.com

[17] "Total Number of Pages Indexed by Google" http://www.statisticbrain.com/total-number-of-pages-indexed-by-google/

[18] "Subject indexing" http://en.wikipedia.org/wiki/Subject_indexing

[19] "Controlled vocabulary" http://en.wikipedia.org/wiki/Controlled_vocabulary

[20] "Spamdexing" http://en.wikipedia.org/wiki/Spamdexing