Reading what machines "*think*"

Fabio Massimo Zanzotto and Danilo Croce

University of Rome Tor Vergata
00133 Roma (Italy)
{zanzotto,croce}@info.uniroma2.it

Abstract. In this paper, we want to farther advance the parallelism between models of the brain and computing machines. We want to apply the same idea underlying neuroimaging techniques to electronic computers. Applying this parallelism, we can address these two questions: (1) how far we can go with neuroimaging in understanding human mind? (*foundational perspective*); (2) can we understand what computers "*think*"? (*applicative perspective*). Our experiments demonstrate that it is possible to believe that both questions have positive answers.

1 Introduction

Studies of machines and of living organisms are strongly related. Biological principles have been often used to design machines. In cybernetics [1], artificial adaptive machines, i.e., machines that can control their states, have been studied with respect to the adaptivity of natural living organisms. This relation between studies of living organisms and design of machines is even more strict when we observe computing machines. It is not hard to imagine that the Von Neumann architecture [2] and the neural-based computing architecture originally introduced by Turing [3] have been inspired by concepts coming from the study of the mind and of the brain. Viceversa, also theories of computer architectures inspired some studies of the mind and the brain. Cognitive Psychology [4] uses computing machines as a metaphor for defining models of the human mind. Cognitive Science (see [5]) is even more radical as computing machines are exploited to test and validate models of the mind.

The strict relation between models of the mind and computing machines is fascinating. Staying in this tradition, in this paper, we want to farther advance this parallelism.

Brains are often studied using neuroimaging techniques to discover areas related to particular cognitive processes. Neuroimaging techniques are also used to induce activation patterns for high-level cognitive processes related to specific semantic categories [6]. These activation patterns can be used to determine what cognitive process a brain is performing.

Computing machines nowadays are extremely complex. These machines perform complex tasks. These tasks seem to be "*cognitive processes*"[1]. For example, manipulating symbols can be seen as a "*cognitive process*".

[1] We often use terms in wrong contexts, e.g., "*cognitive processes*" as related to machines. Yet, we need to make use of human centered terms for describing machines.

We have then a wonderful opportunity. We can think to observe electronic computers with techniques similar to brainimaging. We have good reasons for doing this as we can address these two questions: (1) how far we can go with neuroimaging in understanding human mind? (*foundational perspective*); (2) can we understand what computers "*think*"? (*applicative perspective*).

This is a fascinating research program and in this introduction we gave only a taste of it. We will better describe our vision in Sec. 2. We describe the parallelism between brains and machines that gives the possibility of applying "*brainimaging*" techniques for electronic computers. We better introduce the two reasons motivating this novel view: a *foundational perspective* and an *applicative perspective*. The rest of the paper is organized as follows. In Sec. 3, we report on the studies that are the background for this research. In Sec. 4, we describe the approximated model for studying the parallelism between brains and machines. In Sec. 5, we report on the experiments for testing the approximated model. We here define a test set that can be used for further experiments. Finally, in Sec. 6, we draw some conclusions on this experience and we plan the future work.

2 The vision

Electronic computers nowadays are extremely complex information processing systems. In some sense, these machines are performing "*cognitive processes*". As previously happened in cognitive science and in cognitive psychology studies, we can imagine the parallelism between computers and minds in the field of *neuroimaging*. Computers as well as brains are the physical objects performing "*cognitive processes*". We hereafter call them "*cognitive physical objects*". As shown in Fig. 1, cognitive tasks activate the "*cognitive physical objects*". In both cases, it is possible to observe the activation of these cognitive objects by taking activation images. We can take these activation images by observing different physical phenomena, e.g., electric or magnetic. The parallelism is now complete. On the *brain* side, we have the brain as the observed *cognitive physical object*, a real cognitive task, and classical brainimaging techniques, i.e., fMRI, as the way of observing the brain activation. On the *electronic computer* side, we have the computer as the *observed cognitive physical object*, a program as the *cognitive task*, and images of the electrical activation of micro-chips as a way of observing the computer activation.

The parallelism we made between brains and computers in the field of neuroimaging opens two possible very interesting research perspectives:

- *foundational perspective*: how far we can go with neuroimaging in understanding human mind?
- *applicative perspective*: can we understand what computers "*think*"?

Both research questions are extremely fascinating.

Whenever we misuse a term, we will indicate it with different characters as we did for "*think*" in the title and for "*cognitive processes*".

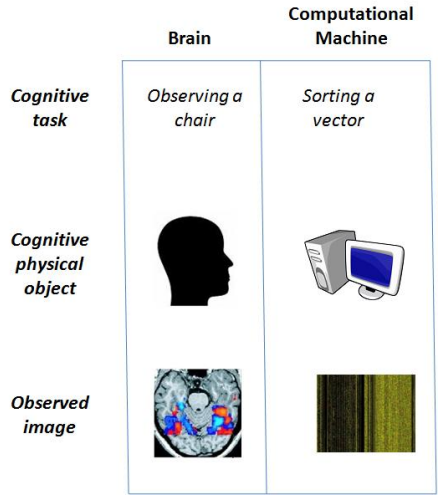

Fig. 1. Observing brains and machines

The *foundational perspective* is extremely important. The aim of some studies in neuroimaging [6, 7] is to determine the correlation between high-level cognitive processes and neuroimages. The idea is that processing different conceptual knowledge produces different brain images representing different activation patterns. For example, different conceptual information such as faces, chairs, and houses produces different activation patterns [6]. These correlations can be used for better understanding the way the human brain organizes conceptual knowledge. Yet, an extremely important question is: *how far we can go with neuroimaging in understanding human mind using these methods?*. With the parallelism between brains and machines we have a wonderful opportunity to answer to this question. On the *brain side* (Fig. 1), we have two known variables, i.e., the required cognitive activity and the observed activation pattern, and one unknown variable, i.e., the way the brain is performing the cognitive process. In brain imaging, the aim is to understand and to model the unknown variable. On the *electronic computer* side, there are no unknown variables: the three elements are completely known. This gives a very relevant test-bed. We know exactly how "*knowledge*" is processed in computers and we know exactly the "*cognitive process*" we ask machines to do. If we succeed in studying the correlation between the cognitive process and the activation image in the electronic computer side, we can be confident that the same method can be used on the brain side. We can also answer two additional questions. On the electronic computer side, we can study if better activation image interpretation models produce better correlations between activation images and cognitive activities. For example, what is the effect of knowing that processes are stored as code and data? Does it help in determining the correlation between activation images and the process "*cog-*

nitive" activity? Answering these kinds of questions on the *electronic computer* side can help in determining if clearer separations between brain images related to different cognitive activities correlate with better understanding of the brain cognitive processes.

The *applicative perspective* is also an extremely interesting and unexplored area of research. Using the ideas developed on the *brain* side of the parallelism (Fig. 1), we can try to apply them to the *electronic computer* side. Can we develop technologies that "*read the computer mind*". This predictive model can have a wide variety of applications, e.g., detecting malicious software, detecting the intentions of hostile computers by looking at their activation patterns. We need specific devices that can capture activation images of computers. We can then study the application of machine learning to induce models that can predict what a computer is doing by analyzing its activation patterns.

The complete realization of the *electronic computer* side of the vision is a long term goal. It requires physical devices to capture the activation state of electronic computers. Yet, as electronic computers are easily and directly observable, it is possible to set up a scenario where we can test the idea. This scenario can help in preparing the ground of the complete research program. This first phase of analysis is the *virtual observation of electronic computers*. We exploit the fact that we can directly observe the memory state of machines and, then, we can draw their activation state. As the observation of the activation state is done through a software system instead of a physical device, we call it *virtual observation*. We will describe this scenario in Sec. 4.

3 Background

Categorization is the cognitive ability of classifying objects into concepts. This ability is extremely important for this study as, on both the *brain* side and the *electronic computer* side, we want to study the correlation between activation images (i.e., objects) and cognitive processes (i.e., concepts). The final aim is to develop models that determine the performed cognitive process by observing an activation image. For example, we want to have a model that determines that the brain in Fig. 1 is performing the act of looking at a chair. This should be done only by observing the brain image.

One of the objectives of machine learning is to define models and algorithms that can learn categorization functions from existing training data. Observing some brain images grouped into classes, i.e., grouped according to the cognitive process, machine learning algorithms induce classifiers that can predict the class for a new and unseen brain image. A classification function C is defined as:

$$C : I \rightarrow T \tag{1}$$

where I is an instance space and T is the set of possible categories. This classification function will observe objects $i \in I$ assigning a class $t \in T$. The categorization is possible if some regularities appear in the space of the instances I. To discover these regularities, we need to observe instances using some feature.

These instances are then represented as points in feature spaces $F_1 \times ... \times F_n$ where each F_i is an observable feature. We can then define a function F that maps instances i in I to points in the feature space, i.e.

$$F(i) = (f_1, ..., f_n) \tag{2}$$

This model is generally called feature-value vector and underlies many algorithms.

The field of machine learning has delivered a wide range of algorithms to analyze huge amounts of data in order to find regularities. Supervised (e.g., [8, 9]), semi-supervised (e.g., [10]), and weakly supervised (e.g., [11]) algorithms and models are available to automatically learn classifiers or decision making systems. These models are widely and successfully applied in many important fields, e.g., homeland security (e.g. [12]), data mining for business intelligence (e.g. [13]), and computational linguistics [14].

Machine learning algorithms have been used to discover regularities in images of brains performing cognitive and semantic tasks. The work in [7] follows the idea that it is possible to discover regularities in brain images of individuals observing or thinking of objects in the same semantic class such as chairs, houses, etc. (e.g., [6]). Machine learning has been applied to induce brain activation patterns for words where the activation image is not observed. Words with similar meaning should have similar activation patterns. Using corpus linguistics, word similarity is determined comparing their *distributional meaning*, i.e., their vectors of co-occurring words. This is the distributional way of determining the meaning of a word [15]. The induced activation patterns have high predictive performance.

4 Virtual Observation of Computational Machines

Electronic computers have a very nice property with respect to our research program. The activity of these machines is observable using software programs. Then, we can simulate the *electronic computer side* of our vision without actually having a physical device to observe the activation state of machines. We can write a software program that snapshots the memory of the machine. These snapshots can then be used to produce activation images as if these were taken from an external device.

Using these virtual observations of the activation states, we can test the overall process of the *electronic computer side*. Then, we can study if it is possible to derive a correlation between the images of the activation states and the performed "*cognitive processes*". For this purpose, we will extract features from activation images to feed machine learning algorithms. Given a set of training examples, i.e., training activation states, associated with different types of "*cognitive activities*", the machine learning algorithm can extract prototypical models of activation for these types of cognitive activities. These latter models can be used to classify novel activation states, i.e., recognize the type of cognitive process that the activation state suggests. If classifiers have good performances with respect to a set of testing activation states, we can conclude that the task

of reading "*machines' thoughts*" is reachable using the proposed features. Finally, we can repeat the process using smoothed images of the activation states. Smoothed images better approximate the images produced with physical observation devices. Then, we can determine if the final vision is viable.

In the rest of this section, we first describe the way of capturing the activation state of machines using software programs (Sec. 4.1). Then, we describe the standard features for image classification we used (Sec. 4.2).

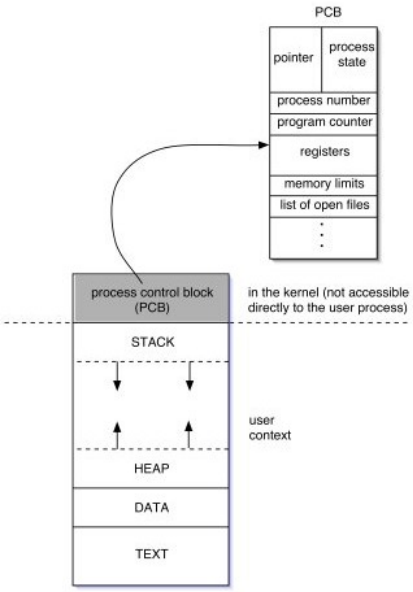

Fig. 2. Organization of the memory for a process

4.1 Capturing the Activation State

In an electronic computer, we can simulate the capturing of the activation state by directly observing the status of the memory. The way we are doing this then is simple. The aim of this phase is to produce an image representing the activation state of a machine performing a particular "*cognitive task*", e.g., sorting a vector or comparing two strings. We exploit the fact that processes perform "*cognitive activities*". We can define here a "*cognitive activity*" as the execution of a program over input data. Processes are completely represented in memory, i.e., both programs and data are stored in memory. Then, we can directly take snapshots of the memory associated with target processes. These snapshots can be used to build images.

Given a cognitive activity, the procedure for extracting images from this activity is then the following:

- running the process representing the cognitive activity, i.e., the program and the related input data
- stopping the process at given states or at given time intervals τ
- dumping the memory associated with the process
- given a fixed height image and the memory dump, read incrementally bytes of the memory dump and fill the associated RGB pixel with the read values
- eventually, produce a smoothed image

This simple procedure can produce more images for each process related to a cognitive activity.

As it is important to explain which part we are using, here we briefly describe the organization of the process memory (see Fig. 2). The process memory contains: the process control block, the stack, the heap, the data, and the program text. The process control block (PCB) contains the information about its status, i.e., the process identification number, the program counter, the registers, the list of opened files, etc. The stack contains information regarding function calls, passed arguments, and local variables. The heap contains dynamically allocated data, e.g., vectors with a length decided at run-time. The data area contains the statically allocated data, e.g., vectors with fixed length decided at compilation time. Finally, the text area contains the compiled program.

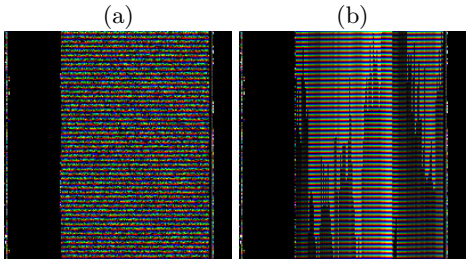

Fig. 3. Sorting Process: two activation states

The relative size of the different memory areas is extremely important. The PCB is extremely small with respect to the other areas. The fact that it is not directly observable from an external process is then not relevant. The information loss can be ignored. The other four areas are instead observable. Yet, in general, the data areas (the heap and the data area itself) are bigger than the code area. Thus, a large part of the memory image represents the data areas.

The process memory is then transformed into an image using the following procedure. Let $M(p)$ be the memory dump of the process p. The memory dump is a sequence of bytes, i.e., $M(p) = [b_0, ..., b_m]$. Using this sequence of bytes we

can produce an image $I(p)$ in Red-Green-Blue (RGB) coding. The image $I(p)$ is a bi-dimensional array of pixels $p_{i,j}$. Each pixel is three contiguous bytes of the memory image. Given the height h of the image, each RGB pixel has the following RGB values:

$$p_{i,j} = [b_{3(i+h\cdot j)}\ b_{3(i+h\cdot j)+1}\ b_{3(i+h\cdot j)+2}] \tag{3}$$

where the first byte $b_{3(i+h\cdot j)}$ is used for the red component, the second $b_{3(i+h\cdot j)+1}$ for the green one, and the third $b_{3(i+h\cdot j)+2}$ for the blue one. Figure 3 provides an example of two memory images for the "*cognitive task*" of sorting a vector. Figure 3(a) is the process at the initial state and Figure 3(b) is the process that accomplished the task. The smaller stripe on the left of each figure is the code area. This is stable during the process execution. The bigger stripe on the right is the vector. At the beginning of the process (Fig. 3(a)), this area is homogeneous as it shows a random vector. At the end of the process (see Fig. 3(b)), we can see a figure suggesting we have a sorted vector.

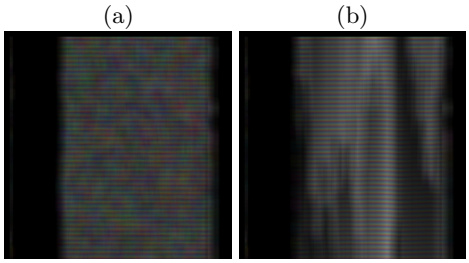

Fig. 4. Sorting Process: two distorted activation states

Finally, to approximate the physical extraction of the activation images, we use a distortion process for the images. This distortion process allows seeing images where contiguous pixels are merged. This approximates the condition of images captured by a physical scanning device. We cannot expect images of the resolutions given by equation (3). This distortion is called *smoothing* or *blurring*. We use the simplest smoothing model, i.e., the rectangular uniform filter. According to this filter, each pixel in the smoothed image $s_{i,j}$ is a weighted sum of the rectangle $n \times m$ of pixels around the target pixel in the original image. If we use $K = n = m$, $s_{i,j}$ are the smoothed image pixels, and $p_{i,j}$ are the original image pixels, the smoothed pixels are defined as:

$$s_{i,j} = \frac{1}{K^2} \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} p_{i+u-\lfloor \frac{K}{2} \rfloor, j+v-\lfloor \frac{K}{2} \rfloor} \tag{4}$$

This kind of distortion is very interesting as it mixes information extracted from contiguous pixels. For example, the distortion of the images in Fig. 3 is reported

in Fig. 4. These latter are obtained using a parameter $K = 10$. This smoothing is particularly relevant as it models what can happen in the situation we have in the final setting, i.e., we are using an external capturing device to extract activation images of electronic computers.

4.2 Feature Space for Images

Once we have the complete or the smoothed activation images, we can model them in the selected feature space to finally use the machine learning algorithm to induce the classifiers. We describe here the features we used. The basic idea is to use what is already available for image processing to estimate how far we can go.

We used three major classes of features: chromatic, textures (OP - OGD) and transformation features (OGD), as described in [16]. Chromatic features express the color properties of the image. They determine, in particular, an n-dimensional vector representation of the 2D chromaticity histograms. Texture features emphasize the background properties and their composition. Texture feature extraction, in LTI-Lib [17], uses the steerability property of the oriented gaussian derivatives (OGD) to also generate rotation invariant feature vectors. Transformations are thus modeled by the OGD features. A more detailed discussion of the theoretical and methodological aspects behind each feature set are presented in [16].

5 Experimental Evaluation

We have now the possibility of investigating whether or not we can solve the problem of determining the "*cognitive process*" given the activation image. This can help in finding initial answers to the questions issued in Sec. 2 related to the *foundational perspective* and the *applicative perspective*.

The rest of the section is organized as follows. First, we describe the experimental settings (Sec. 5.1). Second, we give the results of the experiments (Sec. 5.2). Finally, we discuss these results (Sec. 5.3).

5.1 Experimental setting

For our experiments, we selected 3 different "*cognitive tasks*", i.e., 3 types of algorithms: sorting, comparing two strings, and visiting a binary tree. We used *quicksort* as the sorting algorithm and we selected the *levenshtein distance algorithm* for comparing two strings. We implemented these algorithms in 3 different programming languages (c, java, and php) on a linux platform. We have then 3 different "*cognitive tasks*".

For each of the 3 algorithms in the 3 programming languages, we randomly generated 20 different input data according to the type of algorithm. For the quicksort algorithm, we generated 20 unsorted vectors of 10^5 elements. For the levenshtein distance algorithm, we generated randomly 20 pairs of strings of

100 characters. For the tree visiting algorithm, we generated 20 trees with 10^4 nodes and a random node for each tree. We then have 180 different algorithm-data pairs. For each of the algorithm-data pair, we took 3 snapshots: one at the beginning of the execution, one in the middle, and one at the end. We have then 540 instances. We randomly selected 50% of the instances as training and 50% of the instances for testing. We kept the same distribution of the algorithms and of the languages for both training and testing.

We have then defined two classification tasks:

- *per-algorithm task* (*algo*), the final classes are the 3 different algorithms (i.e., sorting, comparing, and visiting), regardless of the programming language of the implementation
- *per-language task* (*lang*), the final classes are the 3 different programming languages (i.e., c, java, and php), regardless of the implemented algorithm

The *per-algorithm task* is our final task, i.e., understanding whether or not it is possible to determine the "*cognitive task*" performed by the machine. The *per-language task* is instead a control test. We want to see if it is possible to determine the "*cognitive substrate*" where the "*cognitive task*" is performed. As this task seems to be easier and it is similar to the *per-algorithm task*, we want also to see if it is solvable with the feature space for images we are using.

For the two classification tasks, we prepared two different settings according to the smoothing applied to the final images:

- *no-smoothing*, images are kept with the highest quality available;
- *smoothed*, images are smoothed according to the equation (4).

We need these two settings to determine if the performance of the two classification tasks is affected by a degradation of the images. As we already discussed, the degradation approximates the real operational conditions where the captured image cannot have the quality of the single byte of memory.

Finally, as machine learning algorithms we used a decision tree learner [8] ($DecTree$) and a probabilistic model, i.e., the naive bayes ($NaiveBayes$). We selected these two types of algorithms because they behave completely differently. The decision tree learning algorithm recursively selects features. At each step, the one selected is the best discriminatory one. The pruning done in the J48 algorithm also performs a feature selection. Yet, the naive bayes algorithm uses and *weights* all the features in the probabilistic model. These two algorithms then give the possibility of analyzing performances in two completely different setting. We used the implementation given in [18].

5.2 Results

The results of the experiments are reported in Tab. 1. The first column describes the tasks that have been analyzed. The second column reports if the smoothing has been applied. The third column reports the accuracy obtained using the decision tree learning algorithm. Finally, the third column reports the accuracy obtained using a naive bayes probabilistic learning algorithm.

Task	*Smoothing*	*DecTree*	*NaiveBayes*
algo	no	80.37	64.44
lang	no	98.89	99.25
algo	yes	81.48	62.96
lang	yes	98.89	99.62

Table 1. Classification accuracy over the different experimental settings

5.3 Discussion

As expected, the task of deciding the language of the process is simple and solvable. Accuracies are extremely high. There is no difference in performance either between the machine learning algorithms or between the quality of the images. Programming languages produce different organizations of the memory of the processes.

Deciding the algorithm performed by a process is instead more complex. Yet, results are encouraging. The performance of the decision tree learning algorithm is above 80%. The performance is even higher for smoothed images. The differences between the naive bayes algorithm and the decision tree learning suggests that some features are more informative than others. These features are even more important for smoothed images. This increase in performance is unexpected. The behavior of the naive bayes algorithm is instead more predictable as classifiers learnt and applied on plain images performed better than those learnt and applied on smoothed images.

These are initial answers to the two questions issued in Sec. 2. As discussed in the next section, we still need to investigate more complex datasets to generilze these initial findings.

6 Conclusions and future work

We introduced a new vision that can both help in answering questions in neuroimaging and produce novel applications in the computer field. The results of the experimental evaluation suggest we can read what computers "*think*" as we can positively predict "*cognitive processes*" from activation images. Then, we can have positive expectations of the foundational perspective and the application perspective. Yet, the research program we started is still at the beginning and many questions still have to be addressed. For this study we investigated simple "*cognitive processes*". We need to scale-up to different datasets, e.g., datasets containing activation images of word processors and image processors. Then, we have to attack the problem of determining which processes are active in a given memory. Finally, we have to figure out physical devices to directly capture activation images from the electronic computer.

Acknowledgement

We would like to thank Marco Cesati for his precious advices on the organization of the memory in the Linux kernel.

References

1. Wiener, N.: Cybernetics: Or the Control and Communication in the Animal and the Machine. MIT Press, Cambridge, MA (1948)
2. von Neumann, J.: First draft of a report on the edvac. IEEE Ann. Hist. Comput. **15**(4) (1993) 27–75
3. Turing, A.: Intelligent Machinery. In Meltzer, B., Michie, D., eds.: Machine Intelligence. Volume 5. Edinburgh University Press, Edinburgh (1969) 3–23
4. Neisser, U.: Cognitive Psychology. Appleton-Century-Crofts, New York (1967)
5. Von Eckardt, B.: What is cognitive science? MIT Press, Cambridge, MA, USA (1993)
6. Ishai, A., Ungerleider, L.G., Martin, A., Schouten, J.L., Haxby, J.V.: Distributed representation of objects in the human ventral visual pathway. Proc Natl Acad Sci U S A **96**(16) (1999) 9379–84
7. Mitchell, T.M., Shinkareva, S.V., Carlson, A., Chang, K.M., Malave, V.L., Mason, R.A., Just, M.A.: Predicting human brain activity associated with the meanings of nouns. Science **320**(5880) (May 2008) 1191–1195
8. Quinlan, J.: C4:5:programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
9. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1995)
10. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT: Proceedings of the Conference on Computational Learning Theory. Morgan Kaufmann (1998)
11. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B **39**(1) (1977) 1–38
12. Goan, T., Mayk, I.: Improving information exchange and coordination amongst homeland security organizations. In: Proceedings of the 10th International Command and Control Research and Technology Symposium (ICCRTS). (2005)
13. Bose, I., Mahapatra, R.K.: Business data mining : a machine learning perspective. Information & management **39** (2001) 211–225
14. Zanzotto, F.M., Moschitti, A.: Automatic learning of textual entailments with cross-pair similarities. In: Proceedings of the 21st Coling and 44th ACL, Sydney, Australia (July 2006) 401–408
15. Harris, Z.: Distributional structure. In Katz, J.J., Fodor, J.A., eds.: The Philosophy of Linguistics. Oxford University Press, New York (1964)
16. Alvarado, P., Doerfler, P., Wickel, J.: Axon^2 - a visual object recognition system for non-rigid objects. In: IASTED International Conference-Signal Processing, Pattern Recognition and Applications (SPPRA), Rhodes, IASTED (July 2001) 235–240
17. Alvarado, P., Doerfler, P.: LTI-Lib - A C++ Open Source Computer Vision Library. In Kraiss, K.F., ed.: Advanced Man-Machine Interaction. Fundamentals and Implementation. Springer Verlag, Dordrecht (2006) 399–421
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, Chicago, IL (1999)