

Accelerating Semantic Search with Application of Specific Platforms

Marius Monton, Jordi Carrabina, Carlos Montero, Javier Serrano	Xavier Binefa, Ciro Gracia,	Mercedes Blázquez, Jesús Contreras	Emma Teodoro, Núria Casellas, Joan-Josep Vallbé, Marta Poblet, Pompeu Casanovas
Digital Video Understanding Laboratory for HW/SW Prototypes and Solutions (CEPHIS) ETSE-UAB Spain +34935813534	Dept. Computer Science UAB Spain +349358XXXXX	iSOCO (Intelligent Software Components, S.A) +3491XXXXXXX	Institute of Law and Technology (IDT) UAB, Spain +349358XXXXX
{name.surname}@uab.cat	{xavier.binefa; ciro.gracia}@uab.cat	{mercedes;jcontreras}@isoco.com	{name.surname}@uab.cat

ABSTRACT

Semantic Search and Ontologies are one of the key technologies that can improve content management. Nonetheless, in order to be widely diffused, these technologies lack real-time capabilities, that speed up both the indexing and the retrieval processes. This contribution presents the approach and strategy proposed to tackle this problem, within the Spanish project E-Sentencias; a project for the development of a management system for lawyers that includes documentation and multimedia related to the management of their legal cases.

Categories and Subject Descriptors

D.3.3 [Programming Languages]: Language Constructs and Features – *abstract data types, polymorphism, control structures*. This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: <http://www.acm.org/class/1998/>

General Terms

Algorithms, Performance, Design, Experimentation, Theory.

Keywords

Semantic Search, ontology, HW/SW acceleration platforms, Reconfigurable Devices

1. INTRODUCTION

Document indexation has been, up to now, the most commercially reliable and secure approach for the identification and storage of legal documents (in databases) for subsequent retrieval. Nonetheless, there are great disadvantages to this approach, such as the need to index manually, its slowness and subjectivity [1]. Legal professionals spend an important part of their time searching and retrieving specific legal information and, thus, improving the functionalities for search and retrieval of legal documents is paramount for the development of search engines for the legal domain. Moreover, at the moment, all the

information which is available for retrieval is based on text and does not include multimedia files¹

To solve this problem, the E-Sentencias Project will develop a software-hardware system for lawyers to manage the documentation connected to their legal cases and the related multimedia files. Both an ontology-based metasearch engine and a specific hardware platform will be developed to optimize the knowledge generation and management processes in the judicial field. The objectives of this approach are: (i) to save time to users; (ii) to aid searches intelligently; (iii) to optimize the results; and (iv) to improve the organization of the search memory. To achieve these results, the creation of specific legal domain ontologies and the refinement and integration of different existing technologies is needed.

The legal field constitutes a privileged domain for the application of the Semantic Web and several legal ontologies² are being used to construct tools and prototypes to support the management, organization, search and retrieval of documents stored in legal databases [12]. Semantic search, as opposed to keyword search, not only allows information management and retrieval but also knowledge management and retrieval, as it offers the possibility to distinguish between the different meanings contained in a text (or in multimedia files).

¹ According to the Civil Procedure Act (Ley 1/2000, de 7 de enero, de Enjuiciamiento Civil), all civil cases should be recorded in video.

² For more information regarding legal ontologies see [2,3,4,5,6,7,8,9,10,11]

Therefore, the developments regarding ontology-based semantic search offer encouraging qualitative results, as they reduce significantly the amount of information retrieved (compared to indexing) and, at the same time, they improve the quality of the document retrieval process (it filters the non-semantically related documents).

However, quantitatively, semantic searches are not yet sufficiently efficient. As an example, an ontology may be represented as a graph not totally connected (connectivity depends on the domain of application). In order to obtain the maximum similarity/likeness between concepts we need to cover and calculate the distances within this graph. Therefore, the semantic relationships between ontological concepts are equivalent to the distances within the graph. For that reason, the problem becomes an iterative problem from a computational point of view, which allows the application of mathematical classic techniques (Dijkstra's algorithm).

The process to cover an ontology based on 10.000 nodes (with different connectivity degrees) might take from 9 hours up to 4 days. The improvement of the computational time would result in a more efficient and cheaper application of semantic technologies. In this paper we outline the development of the computational acceleration for ontological searches using application specific embedded systems, based on hardware platforms and FPGAs (Field Programmable Gate Arrays) or CPLDs (Complex Programmable Logic Devices).

2. COMPUTATIONAL COMPLEXITY OF IMPLEMENTATION ALGORITHMS

To find out what concepts are semantically related and the quantification of the relationship they have, an appropriate algorithm that takes all possible paths between the two concepts, which calculates scores for all paths and chooses the maximum score, is needed.

This algorithm is a variant of Dijkstra's algorithm [13]. Basic implementations of this algorithm have a computational complexity of $O(|V|^2)$, being V the number of vertices. This complexity appears due to: (1) the Extract-max function, which returns the next vertex with strong relation (large value on edge) and (2) the operation of updating adjacent vertices to last vertex selected.

We can observe that the total number of operations for updating vertices is only $O(E)$, being E the number of edges among vertices. However, the total number of operations in (1) is still $O(|V|^2)$. Some improvements to this algorithm can reduce complexity to $O(|E| + |V| \log |V|)$ using Fibonacci heap to store the graph itself, but this is only true in case of sparse graphs: graphs with much less edges than $|V|^2$, what can be translated to our problem as having much more concepts than binary relations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference'04, Month 1-2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

```
function Dijkstra(G, w, s)
  for each vertex v in V[G]
    d[v] := infinity
  previous[v] := undefined
  d[s] := 0
  S := empty set
  Q := V[G]
  while Q is not an empty set
    u := Extract_Max(Q)
    S := S union {u}
    for each edge (u,v) outgoing from u
      if d[u] + w(u,v) < d[v]
        d[v] := d[u] + w(u,v)
        previous[v] := u
```

Figure 1: Dijkstra's Algorithm

among them. That condition is actually not valid on our ontologies (where the ratio is around 1 to 10), so other techniques are needed to improve this algorithm.

3. INTRODUCTION TO RECONFIGURABLE DEVICES

The computational platforms used for this kind of applications have evolved through the history, experimenting with different architectures (servers, meshes of processors, clusters, etc.). But complex indexing and search problems represent a broad spectrum of algorithms that combine different computations with specific platform requirements, which demands architectures that support such heterogeneity.

When planning the architecture of a general purpose machine, there is the additional requirement of giving support to a number of different applications with the same platform, leading the whole system to be programmable. Our purpose is the development of a system capable of prototyping the implementation for a specific problem, semantic search, with reconfigurable devices.

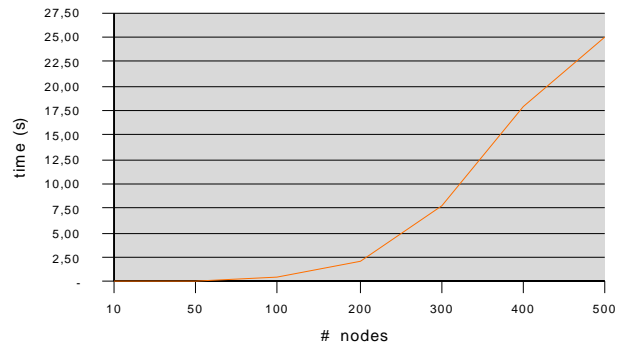


Figure 2: Execution time of the algorithm according to the number of nodes

A system like this is configured in compilation time from a description written in a high level language, partitioned into processes to be executed different resources ranging from processors to application specific hardware resources. This process of describing both hardware and software is based on a set of design methodologies known as “hardware-software co-design”. In last years, they can not only implement hardware modules but also, due to the increasing device size and density, FPGAs can contain several processors (soft core processors).

4. ACCELERATION SCHEME

Our acceleration platform will be based on a PCI expansion card for a standard PC. This PCI card will contain a FPGA as main computational device and the amount of memory required for complex problems. This expansion board will accelerate the process of finding next suitable vertex and maintaining the edges set updated.

Due to amount of memory needed to store all vertices and edges, external memory is used. FPGA will access this memory, and will be accessible to the SW application (usually running in a PC) through a standard Application Programming Interface (API). Using this API, current Dijkstra code (included in the SW application) will call Extract_Max function and this function will be executed on our HW accelerator platform. This way, the most costly function will be implemented in HW, with a fast execution time and running in parallel to the rest of the SW application.

There are three possible approaches to implement the Extract_Max function in our platform using different computational structures: (i) sorted array (ii) sorted linked list and (iii) heap.

4.1 Sorted array

Inside FPGA, it's possible to store small amount of information. The design should allow keeping an array permanently sorted. Insertion is done in the right place, and the rest of the array is shifted accordingly. This implementation is fast (one cycle per

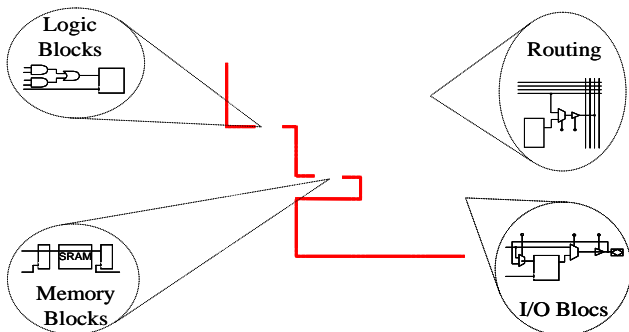


Figure 3: FPGA Basic Blocs

Figure 4: Sorted Array

insertion): its complexity is $O(1)$ for insertions and Extract_Max, but it has great penalty on delete and update functions that are $O(|E|)$, and also in the fact that it cannot store large amounts of data due to internal FPGA memory limitations.

4.2 Sorted linked list

A linked list can be implemented using external memory to FPGA. In this implementation, FPGA is in charge of access to external memory to ensure that the linked list is always sorted. This way, FPGA will find the insertion point by exploring the list and then modify that list to insert, update or delete a node. Complexity of this operation is $O(|E|)$.

4.3 Binary Heap

In this approach, FPGA would be in charge of maintaining a binary heap of nodes. Using this implementation, external memory to the FPGA will be used to store the heap. For every operation, FPGA needs to access several times the external memory, what means slower speed. This implementation has $O(1)$ for Extract_Max function and $O(\log |E|)$ for insert and delete operations, and it can store large amount of edges due use of external memory.

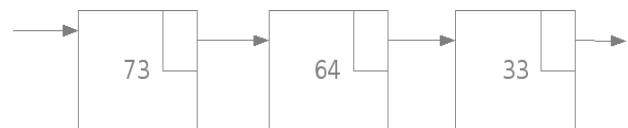


Figure 5: Linked List

(IDT-UAB), Centro de Prototipos y Soluciones Hardware - Software (CHEPIS - UAB) y Digital Video Semantics (Dpt. Computer Science UAB).

Figure 6: Binary Heap

This last solution can be used implement both costly functions Extract-Max function and update adjacent vertices sharing the same storage structure. This method can be optimally implemented with a FPGA and using external memory to store the array of the binary heap.

In these approaches, main bottle-neck will surely be the PCI transfer between PC and our HW acceleration platform. For that reason, next stage of development will involve the implementation of the complete algorithm in the reconfigurable platform, where software only stores graph into memory and PCI board returns all possible relations between nodes.

5. CONCLUSIONS AND FURTHER WORK

In this paper we presented a set of proposals to improve ontology search, based on the implementation of reconfigurable devices. This research is currently being developed within the nationally funded E-Sentencias Project, which has the development of a software-hardware system for lawyers to manage the documentation connected to their legal cases and the related multimedia files as its objective.

With these improvements, we will be able to do complex searches and relationship extractions in large ontologies in few seconds instead of in the current minutes or hours. Moreover, the plan to develop a full platform for managing ontologies can enhance the use of these technologies in new applications.

6. ACKNOWLEDGMENTS

E-Sentencias (E-Sentencias. Plataforma hardware-software de aceleración del proceso de generación y gestión de conocimiento e imágenes para la justicia) is a Project funded by the Ministerio de Industria, Turismo y Comercio (FIT-350101-2006-26). A consortium of: Intelligent Software Components (iSOCO), Wolters Kluwer España, IUAB Institute of Law and Technology

7. REFERENCES

- [1] Casellas, N., Jakulin, A., Vallbé, J.-J. and Casanovas, P. Acquiring an ontology from the text. In M. Ali and R. Dapoigny, editors, *Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2006)*. Annecy, France, June 27-30 2006, Lecture Notes in Computer Science 4031, Springer, 2006: 1000-1013.
- [2] McCarty, L.T. A language for legal discourse, I. Basic features. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*. Vancouver, Canada.
- [3] Stamper, R.K. The role of semantic in legal expert reasoning and legal systems. *Ratio Iuris*, 4(2): 219-244, 1991.
- [4] Stamper, R.K. Signs, Information, Norms and Systems. In B. Holmqvist and P. Andersen, editors, *Signs of Work*. De Gruyter, 1996.
- [5] Valente, A. *A Modelling Approach to Legal Knowledge Engineering*. IOS Press, 1995.
- [6] Breuker, J., Elhag, A., Petkov, E. and Winkels, R. Ontologies for legal information serving and knowledge management. In *Legal Knowledge and Information Systems, Jurix 2002: The Fifteenth Annual Conference*. IOS Press, 2002.
- [7] Kralingen, R.W. van. *Frame-Based Conceptual Models of Statute Law*. Computer/Law Series, No.16, Kluwer Law International, 1995.
- [8] Gangemi, A., Pisanelli, D.M. and Steve, G. A formal ontology framework to represent Norm Dynamics. In *Proceedings of the Second International Workshop on Legal Ontologies*, 2001.
- [9] Casanovas, P., Poblet, M., Casellas, N., Vallbé, J.-J., Ramos, F., Benjamins, V.R., Blázquez, M., Rodrigo, L., Contreras, J. and Gorroñoigoitia, J. *D10.2.1 Legal Case Study: Legal Scenario*. Technical Report SEKT, EU-IST Project IST-2003-506826, 2004.
- [10] Casanovas, P., Casellas, N., Vallbé, J.-J., Poblet, M., Benjamins, V.R. Blázquez, M., Peña-Ortiz, R and Contreras, J. Semantic Web: A Legal Case Study. In J. Davies, R. Studer and P. Warren, editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons, 2006.
- [11] Benjamins, V.R., Casanovas, P., Gangemi, A. and Breuker, J., editors, *Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications*. Lecture Notes in Computer Science. Springer Verlag, 2005.
- [12] Casanovas, P., Casellas, N., Vallbé, J.-J., Poblet, M., Ramos, R., Gorroñoigoitia, J., Contreras, J., Blázquez, M. and Benjamins, V.R.. *Iuriservice II: Ontology Development and Architectural Design*. In *Proceedings of the Tenth*

International Conference on Artificial Intelligence and Law (ICAIL 2005). Alma Mater Studiorum-University of Bologna, CIRSFID, 2005.

[13] Goos, G., Hartmanis, J., and van Leeuwen, J. Intelligent Search on XML Data Applications, Languages, Models, Implementations, and Benchmarks, 125, 2003.