Dynamic Semantic Identification with Complexity Constraints as a Knapsack Problem

M. Falelakis, C. Diou, A. Valsamidis and A. Delopoulos
Multimedia Understanding Group, Information Processing Laboratory
Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki, Greece
Email: {manf, diou, tasos}@olympus.ee.auth.gr, adelo@eng.auth.gr

Abstract— **The process of automatic identification of high level semantic entities (e.g., objects, concepts or events) in multimedia documents requires processing by means of algorithms that are used for feature extraction, i.e. low level information needed for the analysis of these documents at a semantic level. This work copes with the high and often prohibitive computational complexity of this procedure. Emphasis is given to a dynamic scheme that allows for efficient distribution of the available computational resources in application Scenarios that deal with the identification of multiple high level entities with strict simultaneous restrictions, such as real time applications.**

I. Introduction

The extraction of high level *semantic* information from multimedia documents has become a necessity due to the exponential growth of the available multimedia information that calls for efficient indexing and retrieval. Intensive research efforts focusing on the developement of techniques for efficient semantic information extraction are taking place and utilize low level feature extraction algorithms in order to gather information used to identify higher level semantics (see [1] and [2] for example). However such techniques prove to be demanding in terms of computational cost rendering the semantic identification a difficult task, even for modern computers. This problem becomes more apparent in real time applications (e.g., robot navigation and monitoring systems) that deal with identification of multiple semantic entities, often within bulky data sets.

Present work proposes a general method for controlling the complexity of the identification process by evaluating semantic entities partially, in the sense that only a subset of the algorithms that are needed to perform the identification is evaluated. Consequently the overall computational cost is reduced. It turns out that it is possible to select a subset that minimizes the loss in accuracy, effectively providing the best possible results, given the complexity limitations. In section II a knowledge base called the *fuzzy semantic encyclopedia* is presented that is used to define each semantic entity in terms of other lower level semantic entities as well as low level syntactic features. Thus, definitions of this knowledge base correspond to semantic information extraction techniques. Metrics that are used to determine the existence of an entity in a data set; the accuracy, the validity and the complexity of the results are introduced as well. Section III models the problem of finding optimal subsets of algorithms for the identification design as a “knapsack” [3] problem and manages to solve it in pseudo-polynomial time. Another approach that is useful in Scenarios where multiple semantic entities are to be identified is examined in section IV and section V demonstrates design examples providing comparisons between the different design approaches. Finally, section VI concludes this work by commenting on the results and issues that remain to be addressed.

II. Semantic Identification

In this section a knowledge base called *Fuzzy Semantic Encyclopedia* is presented that can be used to cover the so called semantic gap between lower level features and semantic information in a multimedia data set. It consists of definitions of high level *Semantic Entities* in terms of other, lower level Semantic as well as *Syntactic* Entities. Moreover metrics are defined that are used to rank the results and assist in the design of the identification process with limited resources, as explained in the later sections.

A. Syntactic and Semantic Entities

Consider a *syntactic feature* t that is a measurable quantity, such as brightness or frequency, obtained by applying a corresponding algorithm τ on the given data set (e.g., a scene, an image or a signal). A *Syntactic Entity* or property $y_i(t) \in [0, 1]$ is a fuzzy number on a syntactic feature t. For example, the property “very bright” is defined on the feature “brightness” and the property red is defined on the feature “color”. In general, if t_τ is the result of algorithm τ that “measures” t, the membership value $\mu_{Y_i} \equiv y_i(t_\tau)$ corresponds to the *degree* the particular data set assumes property $Y_i \in \mathbf{Y}$, where $\mathbf{Y}$ is the set of all Syntactic Entities considered.

The term *Semantic Entity* refers to higher level objects or concepts that cannot be directly measured and are closer to human perception. The Fuzzy Semantic Encyclopedia is built on the assumption that each Semantic Entity $E_i \in \mathbf{E}$ can be described using other lower level Semantic as well as Syntactic

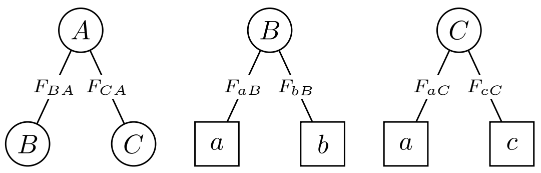


Fig. 1. Definitions of three Semantic Entities in a graph representation. Round and square nodes correspond to Semantic and Sytactic Entities respectively.

Entities that collectively form the *scope* $\mathbf{S}_{E_i}$ of E_i. Any scope is a subset of $\mathbf{S} = \mathbf{Y} \bigcup \mathbf{E}$, the set of all Semantic and Syntactic Entities. If Semantic Entity A is described using the scope $\mathbf{S}_A = \{B, a\}$ while $\mathbf{S}_B = \{a, b\}$, then existence of a and b implies the existence of B and existence of B and a implies the existence of A. However in most cases each Entity $S_i \in \mathbf{S}_E$ implies E *up to a certain degree* $F_{S_i E} \in [0, 1]$. In that sense a *definition* of a Semantic Entity E_k is a discrete fuzzy set of the form

$$E_k = F_{1k}/S_1 + F_{2k}/S_2 + \ldots + F_{nk}/S_n, \tag{1}$$

where $F_{ik} \equiv F_{S_i E_k}$.

A definition that is included in the Encyclopedia is called a *primary definition* and may contain both Semantic and Syntactic Entities. A definition that depends only on Syntactic Entities is called a *detailed definition*. It is shown in [4] that each non-detailed definition can be transformed into a detailed one, so this paper deals only with the latter.

Skipping the formal presentation of this substitution procedure, we illustrate its behaviour by the simple example of Figure 1. The scopes of the Semantic Entities are $\mathbf{S}_A = \{A, B\}$, $\mathbf{S}_B = \{a, b\}$ and $\mathbf{S}_C = \{a, c\}$ and the scope of the detailed definition of A is $\mathbf{S}_{A_d} = (\mathbf{S}_B \bigcup \mathbf{S}_C) \bigcap \mathbf{Y} = \{a, b, c\}$ so we need to calculate the new weights F of the definition $A^d = F^d_{aA}/a + F^d_{bA}/b + F^d_{cA}/c$. We use fuzzy intersection for the transition from b to A via B, hence $F^d_{bA} = \mathcal{I}(F_{bB}, F_{BA})$, where $\mathcal{I}$ is any fuzzy t-norm. Similarly, $F^d_{cA} = \mathcal{I}(F_{cC}, F_{CA})$. There exist two paths that a is related to A, through B with $\mathcal{I}(F_{aB}, F_{BA})$ and through C with $\mathcal{I}(F_{aC}, F_{CA})$. We use fuzzy union to combine these two values, i.e. $F^d_{aA} = \mathcal{U}(\mathcal{I}(F_{aB}, F_{BA}), \mathcal{I}(F_{aC}, F_{CA}))$, where $\mathcal{U}$ is any fuzzy t-conorm. For our purposes the most appropriate fuzzy union and intersection operators are the less “drastic” ones, i.e. the ones that provide granularity in the calculation of the weights F.

B. Metrics

In order to perform the actual identification of a Semantic Entity E_k, i.e. decide whether it exists within a data set, there is a need for a metric that would rank the success of the identification of E_k. In the same manner that the value μ_{Y_i} measures the degree up to which a data set assumes the property Y_i, *Certainty* that E_k exists is given by

$$\mu_{E_k} \triangleq \mathcal{U}_{i}(\mathcal{I}(F_{Y_i E_k}, \mu_{Y_i})) \tag{2}$$

where E_k is a detailed definition.

The maximum possible value of μ_{E_k} is assigned the term *Validity* of the definition and is equal to

$$\mathcal{V}(E_k) \triangleq \mathcal{U}_{i}(F_{Y_i E_k}), \tag{3}$$

attained for $\mu_{Y_i} = 1$ for all Y_i in the scope of E_k and the use of the identity $\mathcal{I}(a, 1) = a$ (true for every t-norm $\mathcal{I}$). Validity is a measure of the information that a definition can provide regarding a Semantic Entity and is therefore a property of the definition itself.

Finally, a definition is characterized by the complexity associated with the algorithms corresponding to its Syntactic Entities. We assign a computational cost $c(t_i)$ to every syntactic feature t and the *Complexity* of a definition is

$$\mathcal{C}(E_k) = \sum_i c(t_i) \tag{4}$$

where each syntactic feature corresponds to a Syntactic Entity $Y_i \in E_k$. Notice that this value will normally depend on the size of the input data, as will the values $c(t_i)$.

III. Optimization Under Validity / Complexity Constraints

Having a limited Complexity budget for the identification of a Semantic Entity E_k one should use only a subset of the Syntactic Entities in $\mathbf{S}_{E_k}$ for the identification. This subset should provide the best possible results for the identification, while the Complexity remains below the required limit.

A. Partial Metrics

Partial Certainty, Validity and Complexity correspond to the metrics presented in section II-B if a subset $\mathbf{A}$ of the Syntactic Entities of a definition E_k is evaluated.

Partial Certainty is defined as

$$\mu_{E_k}(\mathbf{A}) \triangleq \mathcal{U}_{Y_i \in \mathbf{A}}(\mathcal{I}(F_{Y_i E_k}, \mu_{Y_i})) \tag{5}$$

and denotes the confidence we have acquired that E_k exists in a data set by evaluating only the properties in $\mathbf{A}$. In the same manner, partial Validity is

$$\mathcal{V}(E_k/\,\mathbf{A}) \triangleq \mathcal{U}_{Y_i \in \mathbf{A}}(F_{Y_i E_k}). \tag{6}$$

and partial Complexity

$$\mathcal{C}(E_k/\mathbf{A}) \triangleq \sum_{i \in \mathbf{A}} c(t_i) \tag{7}$$

These metrics are nondecreasing and are therefore bounded by the respective total metrics.

B. Design in terms of Complexity and Validity

Given a Complexity threshold C_T for the identification of a Semantic Entity using the detailed definition E_k, we select those subsets $\mathbf{A}_i \subseteq 2^{\mathbf{S}_{E_k}}$ that satisfy the Complexity criterion

$$\mathcal{C}(E_k/\mathbf{A}_i) \leq C_T \tag{8}$$

The optimal subset $\mathbf{A}_T^*$ is the one that maximizes the Validity,

$$\mathbf{A}_T^* = \underset{\mathbf{A}_i}{\text{maximizer}}(\mathcal{V}(E_k/\mathbf{A}_i)) \tag{9}$$

Of course, setting the Complexity threshold $C_T \geq \mathcal{C}(E_k)$ would lead to complete evaluation of the definition.

Similarly if a Validity threshold V_T is given that the identification must exceed, then from all the subsets $\mathbf{A}_i \subseteq 2^{\mathbf{S}_{E_k}}$ that satisfy the Validity criterion

$$\mathcal{V}(E_k/\mathbf{A}_i) \geq V_T \tag{10}$$

the optimal subset is the one that has the smallest Complexity value:

$$\mathbf{A}_T^* = \underset{\mathbf{A}_i}{\text{minimizer}}(\mathcal{C}(E_k/\mathbf{A}_i)). \tag{11}$$

C. Efficient Identification Design

The problem of searching within $|2^{\mathbf{S}_{E_k}}|$ subsets for the optimal one has prohibitive complexity ($\mathcal{O}(2^n)$). In this section this problem is solved as a Knapsack problem [3] with a nonlinear gain function (Validity) in pseudo-polynomial time.

Given a collection $\{1, 2, \ldots, n\}$ of n Syntactic Entities and a Complexity threshold $C_T > 0$, the goal is to find the optimal set $X^*(C_T) = \{x_1, x_2, \ldots, x_n\}$ with $x_i \in \{0, 1\}$ denoting whether the property i has been evaluated ($x_i = 1$) or not ($x_i = 0$), so as to maximize the quantity

$$\overset{n}{\underset{i=1}{\mathcal{U}}}(F_{ik} \cdot x_i) \tag{12}$$

under the restriction

$$\sum_{i=1}^{n} (c(t_i) \cdot x_i) \leq C \tag{13}$$

This problem can be efficiently solved using dynamic programming and this approach solves not only the original problem $X_n(C_T)$, using n algorithms, but also all subproblems of the form $X_j(d)$ where $d = 0, \ldots, C_T$ and $j = \{1\}, \{1, 2\}, \ldots, \{1, \ldots, n\}$ (where the ordering of algorithms plays no role). The optimal solution returns an array containing the solutions $\text{val}^*(d)$, $d = 0, \ldots, C_T$ for all subproblems. Actually $\text{val}^*(d) \equiv \text{val}_n(d)$ where all n algorithms are taken into account. The respective subset is $X^*(d) \equiv X_n(d)$. The Bellman recursion [5] is used where if $\text{val}_{j-1}(d)$ has been computed for all Complexity thresholds $d = 0, \ldots, C_T$ then we can calculate $\text{val}_j(d)$ (i.e. consider algorithm j) as

$$\text{val}_j(d) = \begin{cases} \text{val}_{j-1}(d) & \text{if } d \leq c(j) \\ \max\{val_{j-1}(d), \mathcal{U}(val_j(d - c(j)), F_{jk})\} & \text{if } d \geq c(j) \end{cases} \tag{14}$$

Algorithm 1 Designing with Dynamic Programming.

```
1:  for $d := 0$ to $C_T$ do
2:      $\text{val}_0(d) := 0$
3:  end for
4:  for $j := 1$ to $n$ do
5:      for $d := 0$ to $c(j)$ do
6:          $\text{val}_j(d) := \text{val}_{j-1}(d)$
7:      end for
8:      for $d := c(j)$ to $C_T$ do
9:          if $\mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jk}) > val_{j-1}(d)$ then
10:             $\text{val}_j(d) := \mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jk})$
11:         else
12:             $\text{val}_j(d) := \text{val}_{j-1}(d)$
13:         end if
14:     end for
15: end for
16: for $d := 0$ to $C_T$ do
17:     $\text{val}^*(d) := \text{val}_n(d)$
18: end for
```

The pseudocode of this process is presented in Algorithm 1.

In order to compute the optimal sets of algorithms $X^*(d)$ for each threshold (see [3] pp. 22 - 25) we observe that the set $X_j(d)$ differs from an item of the previous iteration $j - 1$ by at most one element x_j. It is thus sufficient to keep, in every iteration, a pointer $A_j(d) \in [0, 1]$, that is $A_j(d) = 1$ if $\text{val}_j(d) := \mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jk}) > val_{j-1}(d)$ and j was included, while $A_j(d) = 0$ if $\text{val}_j(d) := \text{val}_{j-1}(d)$ and algorithm j was not included. Finally to construct the set $X^*(d)$ by going through the pointers, we do the following: If $A_n(d) = 1$ then algorithm n belongs to the optimal subset and we go on by checking $A_{n-1}(d - c(j))$, otherwise ($A_n(d) = 0$) n was not included and we proceed with $A_{n-1}(d)$.

To prove the correctness of Algorithm 1, we notice that for $j = 1$ the values $\text{val}_1(d)$ are the optimal for any threshold d, since there is only one algorithm which can be included or not. We assume by induction that for $1 < j \leq n$, $\text{val}_{j-1}(d)$ are the optimal Validities for any d, $0 \leq d \leq C_T$. The following cases exist:

1) The available Complexity d is less than the Complexity $c(j)$ so j cannot be included, hence the optimal subset has $\text{val}_j(d) = \text{val}_{j-1}(d)$.
2) There is enough Complexity for j, i.e. $d \geq c(j)$ and j may be included.

For the second case we must prove that $\text{val}_j(d) = \max(\text{val}_{j-1}(d), \mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jk}))$. If j is not included in the subset, then the maximum Validity occurs from the subset calculated in $j - 1$ and is $\text{val}_{j-1}(d)$. If j is included, then j can be added to any subset corresponding to $\text{val}_{j-1}(l)$, where $l \leq d - c(j)$ (since j needs at least $c(j)$ of the available Complexity budget). In this case the maximum possible Validity of the subset is $\text{val}_j(d) = \mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jl})$ because $\text{val}_{j-1}(d - c(j)) \geq \text{val}_{j-1}(l)$ (Validity is nondecreasing). The optimal Validity will therefore occur by the maximum of the two possible values, i.e. $\text{val}_j(d) = \max(\text{val}_{j-1}(d), \mathcal{U}(\text{val}_{j-1}(d - c(j)), F_{jk})$. Hence Equation (14) on which Algorithm 1 is based yields the correct results when Validity is the gain function.

The computational complexity of Algorithm 1 is $\mathcal{O}(nC_T)$, i.e. the problem $X_n(C_T)$ is solved in pseudo-polynomial time. At the same time, all subproblems of the form $X_j(d)$ are solved as well, so setting a Complexity threshold $C_T \geq C_{\text{total}}$ solves all the design problems for a Semantic Entity.

IV. Dynamic Simultaneous Identification

The design scheme presented in the previous section allows for efficient identification of a single Semantic Entity in a document with limitations in the available Complexity. The design takes place before the actual search and is based on Validity, a metric that depends solely on the definition used and is therefore a priori computable.

For real-time environments, where multiple Entities need to be identified, and/or for systems that need to identify Semantic Entities in multiple documents a dynamic scheme that constantly reevaluates the design based on the identification results is more adequate. Consider the example of a traffic monitoring system that uses cameras to monitor a street junction. Semantic Entities such as "Traffic Congestion", "Traffic Jam" and "Accident" need to be identified. The identification has to be performed in real-time while the system resources are limited and exhaustive evaluation of the Syntactic Entities is impossible. In the dynamic scheme proposed in this section, a fraction of the available Complexity "budget" (processor time, for example) is first distributed among the Semantic Entities and the identification is performed with a low Complexity threshold for each one of them. The remaining Complexity is then given to the Entities that acquire a relatively high Certainty i.e. there is an indication that they exist in the data set.

The simplest solution to the problem of sharing the available Complexity C_A is to distribute complexity fractions $g_k \cdot C_A$ to each Semantic Entity E_k, where $\sum_k g_k = 1$ and $0 < g_k < 1$ denotes the importance of the Semantic Entity E_k in the scenario considered. The design is then performed for each Semantic Entity separately, as presented in section III-B. However this approach is not optimal mainly due to the fact that it does not take into account that definitions may have common Syntactic Entities that need to be evaluated only once. In the following we propose two different approaches that can be used to cope with this issue.

A. The "Competitive" Approach

Suppose we need to identify the Semantic Entities $E_i \in \mathbf{E}_s$ with a limitation C_s in the available Complexity. Similarly to the definitions in the Semantic Encyclopedia that use Semantic and Syntactic Entities we may consider E_i as members of the definition of an even higher level Semantic Entity W that is the application scenario i.e.

$$W = G_{1W}/E_1 + G_{2W}/E_2 + \ldots + G_{nW}/E_n, \qquad (15)$$

where the weights G have a similar meaning to the weights F used in a definition, denoting the importance of each Semantic Entity for the scenario. The scope of W is therefore $\mathbf{S}_W \equiv \mathbf{E}_s$.

The definitions E_i are detailed and in order to compute the direct contribution of each Syntactic Entity in the scope $\mathbf{S}_W^d = \mathbf{S}_{E_1} \cup \ldots \cup \mathbf{S}_{E_n}$, we use the same method that has been presented with an example in section II-A. For each Syntactic Entity Y_j participating in one of the definitions E_i we calculate the degree up to which Y_j is related to W as

$$G_{Y_j W} = \mathcal{I}(F_{Y_j E_i}, G_{iW}). \qquad (16)$$

If a Syntactic Entity participates in more than one definitions E_l we use fuzzy union to combine the different values, consequently

$$G_{Y_j W} = \mathop{\mathcal{U}}_{l}(\mathcal{I}(F_{Y_j E_i}, G_{iW})), \qquad (17)$$

for all $l \in \{i \mid Y_j \in \mathbf{S}_{E_i}\}$.

Having obtained the direct relation of each Syntactic Entity to the scenario W it is possible to design the identification with a threshold C_s using the design in terms of Complexity presented above (section III-B). The criterion that needs to be maximized in this case is

$$\mathcal{V}(W) = \mathop{\mathcal{U}}_{j}(G_{Y_j W}). \qquad (18)$$

By using fuzzy union in its definition, maximization of $\mathcal{V}(M)$ becomes essentially equivalent to requiring a good performance of the identification "as a whole" even if some monitored entities "are paid less attention". The design using dynamic programming presented in section III-C applies here as well. Equations (16) and (17) assure that a Syntactic Entity that participates in multiple definitions or is important for the identification of a Semantic Entity in the Scenario will be assigned a high value G.

Note that this process is used to select the Syntactic Entities that will be evaluated during the identification of E_i with limited Complexity and the "Scenario" has no perceptual counterpart, that is, no Certainty can be defined for it. Evaluation of the aforementioned Syntactic Entities will yield Certainty values for each Semantic Entity E_i separately. The selection of Syntactic Entities, however, does not take into account the Semantic Entities in which they participate, but aims at the optimal use of the available Complexity. It is therefore possible that an Entity is completely ignored in the sense that none of its properties are selected, in favor of other Semantic Entities that are defined by properties that have a greater contribution to the Scenario. This justifies the use of the term "competitive" for this approach.

B. The "Welfare" Approach

If a subset of Syntactic Entities participating in a definition is selected, then Validity is a measure of the accuracy of the results obtained by using this subset. If, contrary to the "competitive" approach, we wish to ensure that all the Semantic Entities of the scenario are taken into consideration then the selected subset of Syntactic Entities should maintain decent Validity levels for each Semantic Entity. This can be achieved by selecting a subset that maximizes a function $Z(V_{E_1}, \ldots, V_{E_n})$ that has the following properties:

1) If the Validity of a Semantic Entity decreases then Z is also decreased and if $V_{E_i} = 0$ for a Semantic Entity E_i, then Z becomes zero as well.
2) If V_{E_i} is the minimum of the Validities then $Z \leq V_{E_i}$.

These properties ensure that no Semantic Entity is completely ignored and while no Semantic Entity has a low Validity in favor of another. We observe that every t-norm satisfies the above conditions, but again, the best functions for the design are the ones that provide granularity in the identification (i.e. its value is determined by all arguments, contrary to the case of $\min$ for example, where only the minimum value is taken into account). A reasonable choice is the algebraic product, hence $Z(V_{E_1}, V_{E_2}, \ldots, V_{E_n}) = V_{E_1} V_{E_2} \ldots V_{E_n}$.

Consequently, in this approach (the "welfare" approach) if $\mathbf{S}^d_W$ are the Syntactic Entities participating in the scenario and $2^{\mathbf{S}^d_W}$ are all their possible combinations, then we select those $\mathbf{A}_i \in 2^{\mathbf{S}^d_W}$ that satisfy the Complexity criterion $\mathcal{C}(\mathbf{A}_i) \leq C_s$. From all the sets $\mathbf{A}_i$ we select the one that maximizes Z,

$$\mathbf{A}^* = \text{maximizer}(Z(V_{E_1}, \ldots, V_{E_n})), \tag{19}$$

where $V_{E_i} = \mathcal{V}(E_i / \mathbf{A}^*)$, the Validity of E_i obtained for the set of Syntactic Entities under consideration.

Finding the optimal subset i.e. the one that maximizes the function Z is also a problem with exponential complexity and cannot be directly solved using dynamic programming contrary to the case of designing in terms of Complexity for a single Entity. However a near-optimal solution can be obtained by using a variation of Algorithm 1, presented in Algorithm 2.

Algorithm 2 Designing with Dynamic Programming for the Welfare Approach.

```
1: for $d := 0$ to $C_T$ do
2:     $z_0(d) := 0$
3: end for
4: for $j := 1$ to $n$ do
5:     for $d := 0$ to $c(j)$ do
6:         if $z_j(d-1) < z_{j-1}(d)$ then
7:             $z_j(d) := z_{j-1}(d); algos_j(d) := algos_{j-1}(d)$
8:         else
9:             $z_j(d) := z_j(d-1); algos_j(d) := algos_j(d-1)$
10:        end if
11:    end for
12:    for $d := c(j)$ to $C_T$ do
13:        $A := algos_{j-1}(d - c(j))$
14:        $t := \underset{k}{\mathcal{I}}(\mathcal{U}(\underset{i \in A}{\mathcal{U}}(F_{ik}), F_{jk}))$
15:        if $t > z_{j-1}(d)$ then
16:            if $t > z_j(d-1)$ then
17:                $z_j(d) := t$
18:                $algos_j(d) := \{A, j\}$
19:            else
20:                $z_j(d) := z_j(d-1); algos_j(d) := algos_j(d-1)$
21:            end if
22:        else
23:            $z_j(d) := z_{j-1}(d); algos_j(d) := algos_{j-1}(d)$
24:        end if
25:    end for
26: end for
27: for $d := 0$ to $C_T$ do
28:     $z^*(d) := z_n(d)$
29: end for
```

V. EXPERIMENTS

A set of indicative synthetic experiments was carried out, the results of which are displayed in this section.

During the first experiment, a random detailed definition, consisting of 100 syntactic features with uniformly distributed values for Complexity and weight factors was created. Figure 2 illustrates the attained Validity for various Complexity thresholds. It can be noticed that decent Validity values are obtained under relatively strict Complexity limitations. Increasing the threshold is always followed by an increase of the attained Validity, but after a certaint point, this increase is not proportional to the required computational cost. It is also important to mention that even though such a definition would entail an astronomical cost of 2^{100} computations if calculated exhaustively, it only takes seconds to run on a typical pentium PC, when using Algorithm 1.

In the next experiment, we constructed definitions for three different entities, using 10 algorithms with random Complexity and weight factors and then designed the identification process according to the "competitive" and the "welfare" approach. Figure 3 displays the Validity attained for various Complexity thresholds for each entity. We notice that using a "welfare" approach (Figure 3(a)), results in a more balanced increase of Validities in terms of Complexity spent. On the other hand, the "competitive" approach (Figure 3(b)) achieves higher total Validities much faster.

Finally, figure 4 shows the Z values attained when using Algorithm 2 (dynamic programming) versus the optimal ones resulting from an exhaustive evaluation of all possible subsets. As expected, the results of Algorithm 2 are suboptimal, providing, however, a very satisfactory approximation of the optimal values.

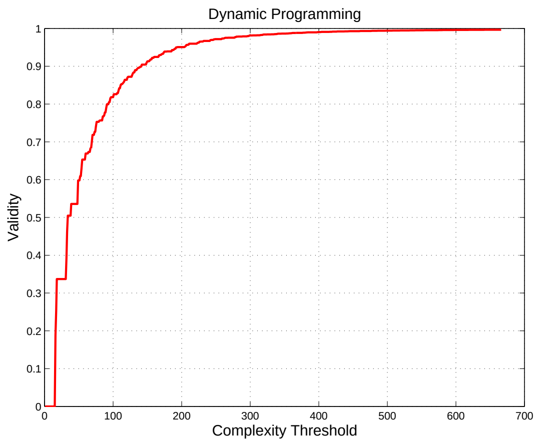


Fig. 2. Designing in terms of Complexity for a single entity using dynamic programming.

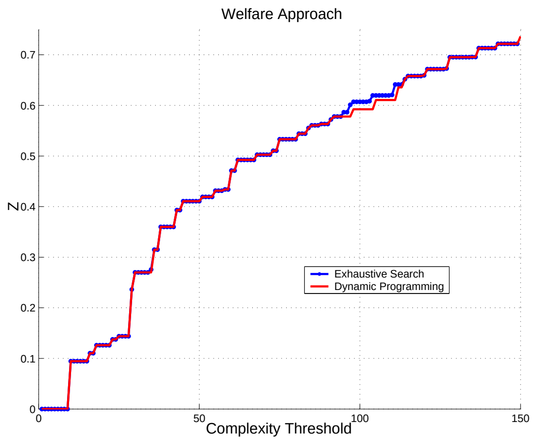
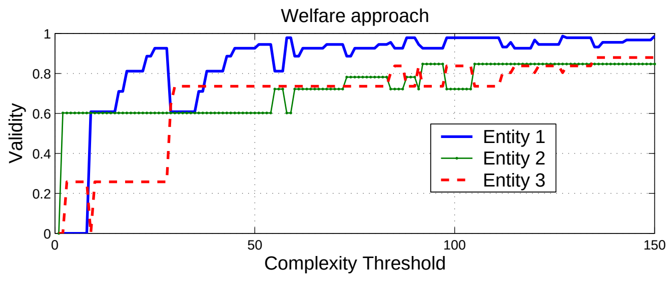
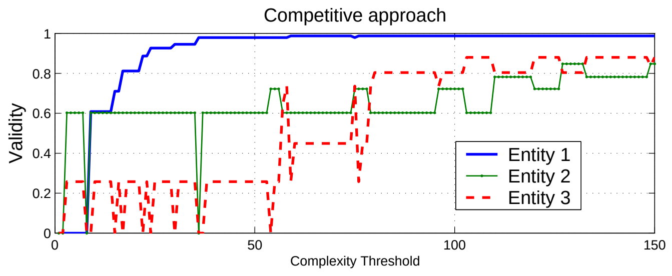


Fig. 4. The welfare approach. Attained z values using exhaustive search vs dynamic programming.



(a)



(b)

Fig. 3. Attained Validity for each semantic entity vs. various Complexity thresholds; (a)the welfare and (b)the competitive approach;.

VI. Conclusions

A set of methods that can achieve Complexity control in the semantic identification was presented in this paper. Three metrics, namely Certainty, Validity and Complexity were defined and allowed the design in terms of Complexity restrictions by using only a subset of the algorithms involved. The straightforward task of selecting the optimal subset of Syntactic Entities to use when designing the identification of a single Semantic Entity has a prohibitive computational cost, a problem that was confronted with the use of dynamic programming.

When dealing with multiple Semantic Entities in a data set two methodologies were proposed for sharing the available computational Complexity. The "welfare" approach aims at the distribution of resources in a way that preserves the balance between the Validities of each individual Semantic Entity. On the other hand, the "competitive" approach makes no provisions for each separate Semantict Entity but tends to increase the overall accuracy of the identification.

Possible applications of the methodologies introduced in this work include systems where strict time/complexity limitations apply for the identification of multiple semantic entities such as robot navigation in unknown environments and traffic monitoring systems. The proposed methodologies could also prove useful in systems dealing with large amounts of data such as medical knowledge bases as well as multimedia indexing and retrieval applications.

References

[1] J. Assfalg, M. Bertini, C. Colombo, and A. Del Bimbo. Semantic annotation of sports videos. *IEEE Multimedia*, 9(2), April-June 2002.

[2] R. Leonardi, P. Migliorati, and M. Prandini. Semantic indexing of sport program sequences by audio-visual analysis. In *IEEE International Conference on Image Processing, ICIP*, Barcelona, Spain, September 2003.

[3] Hans Kelleler, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004.

[4] M. Falelakis, C. Diou, A. Valsamidis, and A. Delopoulos. Complexity control in semantic identifi cation. In *IEEE International Conference on Fuzzy Systems, Reno, Nevada, USA*, May 2005.

[5] R.E Bellman. *Dynamic Programming*. Princeton University Press, 1957.