Clustering and Classification in Structured Data Domains Using Fuzzy Lattice Neurocomputing

Vassilios Petridis, and Vassilis G. Kaburlasos
Aristotle University of Thessaloniki
Dept. Electrical and Computer Engineering
GR-54006 Thessaloniki, Greece

Abstract

A connectionist scheme, namely s- *F*uzzy *L*attice *N*eurocomputing scheme or s-*FLN* for short, which has been introduced in the literature lately for clustering in a lattice data domain, is employed in this work for computing clusters of directed graphs in a master-graph. New tools are presented and used here including a convenient inclusion measure function for clustering graphs. A directed graph is treated by s-*FLN* as a single datum in the mathematical lattice of subgraphs stemming from a master-graph. A series of experiments is detailed where the master-graph emanates from a Thesaurus of a spoken language synonyms. The words of the Thesaurus are fed to s-*FLN* in order to compute clusters of semantically related words, namely *hyperwords*. The arithmetic parameters of s-*FLN* can be adjusted so as to calibrate the total number of hyperwords computed in a specific application. It is demonstrated how the employment of hyperwords implies a reduction, based on the *a priori* knowledge of semantics contained in the Thesaurus, in the number of features to be used for document classification. In a series of comparative experiments for document classification it appears that the proposed method improves favorably classification accuracy in problems involving longer documents whereas performance deteriorates in problems involving short documents.

Keywords: Text Classification, Neural Networks, Clustering, Graphs, Framework of Fuzzy Lattices.

1 Introduction

A sustained interest in *connectionist models*, or, alternatively, *neural networks*, has been revitalized more than ten years ago. Connectionist models have been quite efficient for vector-based or for sequential representations due, among other, to their capacity for generalization, parallel processing, as well as their ability for real function approximation [Scarselli]. Nevertheless, in several applications other types of data than vectors may arise. For instance, terms in first-order logic, blocks in document processing, patterns in

structural and syntactic pattern recognition are entities which are best represented as graph structures and they cannot be easily dealt with vector-based architectures [Sperduti97]. However, the conventional connectionist models, despite their aforementioned qualities, fall short of processing non-numeric data.

Interest in developing connectionist architectures capable of dealing with non-numeric data can be traced back to the end of the 1980's. Early approaches include BoltzCONS [Touretzky], and Recursive Auto-Associative Memory (RAAM) [Pollack]. A widely referenced neural model of learning past tenses of English verbs has also been presented in [Rumelhart]. More recent techniques include labeled RAAM [Sperduti95], and holographic reduced representations [Plate]. Recurrent neural networks have been proposed as connectionist models with a feedback mechanism in their network architecture so as they can be trained to behave like deterministic finite-state automata [Omlin]. Furthermore, the need for a unified treatment of several types of data structures has been acknowledged [Frasconi].

The problem of learning based on "other than fixed-length" feature vectors has been treated in a non-connectionist context in [Cohen] where set-valued features are employed. Nevetheless a major difficulty regarding the employment of connectionist models for non-numeric data processing is in devising proper ways of learning. In typical neural network applications, which involve non-vector data, a suitable vector-based data representation is forced out as an observation [Lecun]. Further data processing may be carried out on a conventional multilayer perceptron architecture [Gori] whose well-known drawbacks include: 1) long training time, 2) "erosion" of previous knowledge when new training data are presented, and 3) no reasons for the answers can be provided. Note that the common practice in neural computing when non-numeric data are involved, that is of converting the non-numeric data to numeric ones, might be regarded as one type of data preprocessing which could potentially suppress original discriminatory features and lead to deterioration in decision making.

A fundamentally novel approach to neurocomputing, namely *F*uzzy *L*attice *N*eurocomputing (*FLN*), has been proposed by the authors of this paper for dealing with non-numeric data without converting them to numeric data [Kaburlasos97], [Petridis98]. More specifically the *FLN* is a connectionist paradigm in the framework of fuzzy lattices (*FL-framework*) which is detailed rigorously in [Petridis99]. An *FLN* scheme implies *versatility* since an *FLN* scheme can be applied, in principle, to disparate data domains which might be any one of the following domains: vectors of numbers, fuzzy sets, symbols, etc. as it has been demonstrated in [Kaburlasos97], [Petridis98], [Petridis99]. A specific *FLN* scheme is shown in this paper, namely *s-FLN* (scheme), inspired from the biologically motivated *A*daptive *R*esonance *T*heory (ART) for neural computation [Carpenter87], [Carpenter91]. The *s-FLN* is employed in this work for computing clusters of directed graphs in a master-graph aiming at document classification based on semantics.

Apart from the treatment of a whole graph as a single datum in neural computing, another suggestion of this work for improving the state-of-the-art involves the application of neural computing for document

classification based on semantics. The latter is effected as described in this work by computing graph clusters in a master-graph stemming from a Thesaurus of linguistic synonyms. Note that various approaches to document classification employ vectors of features, where a feature corresponds to an occurrence of a specific word in a document [Drucker], [Mladenic98], [Sahami]. For a recent survey on intelligent agents for learning text the reader may refer to [Mladenic99]. Lately, the idea of relating documents based on content rather than merely on statistics is "gaining momentum", for example in [Chang] a feedback mechanism is proposed for information retrieval based on content. In the same context, *a priori* knowledge of semantics has been employed as explained in the following. More specifically, the problem of text classification is dealt with in [Junker] by integrating the WordNet Thesaurus in conventional rule induction algorithms; furthermore, in [Green] the WordNet Thesaurus is employed as well to generate automatically links between semantically related documents.

The layout of this paper is as follows. Section 2 delineates the *framework of fuzzy lattices* (*FL-framework*) and it introduces novel tools for clustering graphs. Section 3 introduces lattice G of directed graphs and it illustrates by an example the utility of various tools in lattice G. In section 4 a neural scheme for unsupervised clustering, namely *s-FLN* scheme, is described algorithmically and analyzed. Section 5 details an application of *s-FLN* for graph clustering in a master-graph emanating from MOBY Thesaurus of linguistic synonyms in order to imply a semantics-based dimensionality reduction of the feature vectors to be employed for document classification. A series of comparative experiments for document classification is also reported. Finally, section 6 summarizes the contribution of this work and it discusses the potential utility of the presented techniques.

2 The Framework of Fuzzy Lattices (*FL-framework*): A Theoretical Foundation for Structured Data Processing

In order to enable processing of non-numeric data including graphs, the framework of fuzzy lattices or *FL-framework* for short, has been introduced. This section summarizes briefly useful notions and tools. For a more general and formal introduction to the *FL-framework* the reader may refer to [Petridis99]. An important notion in the *FL-framework* is the notion *partly ordered set* which is defined as follows.

Definition 1 A **partly ordered set** is a set in which a binary relation $x \leq y$ is defined, which satisfies the following conditions for all x,y,z

P1. For all x, $x \leq x$. (Reflexive)

P2. If $x \leq y$ and $y \leq x$, then $x = y$. (Antisymmetry)

P3. If $x \leq y$ and $y \leq z$, then $x \leq z$. (Transitivity)

∇

The definition of the well-known notion *lattice* is cited underneath from [Birkhoff].

Definition 2 A **lattice** is a partly ordered set any two of whose elements have a greatest lower bound or **meet** denoted by $x \wedge y$ and a least upper bound or **join** denoted by $x \vee y$.

∇

A lattice is called *complete* when each of its subsets has a least upper bound and a greatest lower bound in the lattice in question. A nonvoid complete lattice contains a least element and a greatest element denoted by O_{L} and I_{L}, respectively, as suggested in [Petridis99]. Note that only complete lattices will be considered in this work, therefore in the sequel by "lattice" is meant "complete lattice". A *lattice* is called in this work *conventional lattice*, or alternatively, *crisp lattice*. The partial ordering relation in a crisp lattice L will be denoted by $\leq_{\mathsf{L}}$, whereas the join and meet operations in L will be denoted, respectively, by $\vee_{\mathsf{L}}$ and $\wedge_{\mathsf{L}}$.

If $x,y \in \mathsf{L}$ then either "x and y are comparable", that is either $x \leq_{\mathsf{L}} y$ or $y \leq_{\mathsf{L}} x$, or "x and y are incomparable" that is neither $x \leq_{\mathsf{L}} y$ nor $y \leq_{\mathsf{L}} x$. Two incomparable elements $x,y \in \mathsf{L}$ are denoted by $x \|_{\mathsf{L}} y$. Note that the subscript in all of $\leq_{\mathsf{L}}$, $\vee_{\mathsf{L}}$, $\wedge_{\mathsf{L}}$, and $\|_{\mathsf{L}}$ is meant to explicitly identify the underlying lattice. When $\leq$, $\vee$, $\wedge$, and $\|$ are used in this work without a subscript they refer to the set R of real numbers. Note that R is itself a lattice such that for $x,y \in \mathsf{R}$ it holds $x \wedge y = min\{x,y\}$ and $x \vee y = max\{x,y\}$. Furthermore $x \| y$ is always false in R, symbolically $x \not\| y$, hence the elements in R are called *totally ordered* and R is called *totally ordered lattice* or, alternatively, *chain*.

The notion *fuzzy lattice* has been introduced in [Kaburlasos97], [Petridis98], [Petridis99] in order to extend the lattice ordering relation to all pairs $(x,y) \in \mathsf{L} \times \mathsf{L}$ of a crisp lattice L. Such an extended relation may be regarded as a fuzzy set on the universe of discourse $\mathsf{L} \times \mathsf{L}$. In this work a fuzzy set is denoted by (X,μ), where X is the universe of discourse and μ is a membership function $\mu: \mathsf{X} \rightarrow [0,1]$. Hence, the aforementioned extended relation implies a fuzzy set $(\mathsf{L} \times \mathsf{L},\mu)$, which is defined under condition $\mu(x,y)=1$ if and only if $x \leq_{\mathsf{L}} y$. The definition for a fuzzy lattice follows naturally.

Definition 3 A **fuzzy lattice** is a pair $\langle \mathsf{L},\mu \rangle$, where L is a crisp lattice and $(\mathsf{L} \times \mathsf{L},\mu)$ is a fuzzy set such that $\mu(x,y)=1$ if and only if $x \leq_{\mathsf{L}} y$.

∇

The collection of fuzzy lattices is referred to as *framework of fuzzy lattices* or *FL-framework* for short. The significance of the above definition is that it allows one to specify the degree of inclusion of a crisp lattice's element to any other element. Note that $\mu(x,y)=1$ in a fuzzy lattice $\langle \mathsf{L},\mu\rangle$ does not necessarily imply $\mu(y,x)=0$ and it could well be $\mu(y,x)>0$. Regarding *transitivity* in a fuzzy lattice $\langle \mathsf{L},\mu\rangle$ note that the conventional transitivity property holds only in the sense that $\mu(x,y)=1$ and $\mu(y,z)=1$ jointly imply $\mu(x,z)=1$. When it is either $\mu(x,y)\neq 1$ or $\mu(y,z)\neq 1$ then $\mu(x,z)$ could be any number in [0,1]. The following definition for an *inclusion measure* from [Kaburlasos97], [Petridis98], [Petridis99] will eventually enable the fuzzification of a complete crisp lattice.

Definition 4 Let L be a complete lattice. An **inclusion measure** in L is a map $\sigma: \mathsf{L}\times\mathsf{L}\rightarrow[0,1]$ such that $\sigma((x,u))\equiv\sigma(x\leq_{\mathsf{L}} u)$ satisfies the following three conditions:

(*E0*) $\sigma(x\leq_{\mathsf{L}} O_{\mathsf{L}})=0$, $x\neq O_{\mathsf{L}}$, where O_{L} is the least element in L.

(*E1*) $\sigma(u\leq_{\mathsf{L}} u)=1$, $\forall u\in\mathsf{L}$.

(*E2*) $u\leq_{\mathsf{L}} w \Rightarrow \sigma(x\leq_{\mathsf{L}} u)\leq\sigma(x\leq_{\mathsf{L}} w)$, $x,u,w\in\mathsf{L}$ - *Consistency Property*.

∇

It can be argued that $\sigma(x,u)$ indicates the degree of inclusion of x in u, therefore notations $\sigma(x,u)$ and $\sigma(x\leq_{\mathsf{L}} u)$ will be employed interchangeably. To define an inclusion measure in a crisp lattice L a real number will be attached to each of its elements by a *valuation* function. A *valuation* on crisp lattice L is a real-valued function v: $\mathsf{L}\rightarrow\mathsf{R}$ which satisfies $v(x)+v(y)=v(x\vee_{\mathsf{L}} y)+v(x\wedge_{\mathsf{L}} y)$, $x,y\in\mathsf{L}$. A valuation is *monotone* if and only if $x\leq_{\mathsf{L}} y$ implies $v(x)\leq v(y)$, and *positive* if and only if $x<_{\mathsf{L}} y$ implies $v(x)<v(y)$ [Birkhoff]. A positive valuation v on a lattice L renders the lattice in question a *metric space* with metric (distance) : $d(x,y)= v(x\vee_{\mathsf{L}} y)-v(x\wedge_{\mathsf{L}} y)$, $x,y\in\mathsf{L}$ [Birkhoff]. It is assumed here that $v(O_{\mathsf{L}})=0$ for a positive valuation function because if $v(O_{\mathsf{L}})\neq 0$ then another positive valuation function v^{+} with $v^{+}(O_{\mathsf{L}})=0$ can always be defined out of v by subtracting $v(O_{\mathsf{L}})$ from all $v(x)$, $x\in\mathsf{L}$. The following theorem for defining an inclusion measure in L is from [Petridis99].

Theorem 5 The existence of a positive valuation function v on a crisp lattice L is a sufficient condition for function $k(x,u)=\dfrac{v(u)}{v(x\vee_{\mathsf{L}} u)}$ to be an inclusion measure in lattice L.

∇

Due to the application requirements in the context of this work another inclusion measure will be defined next.

Theorem 6 The existence of a positive valuation function v in a crisp lattice L is a sufficient condition for function $s(x,u)=\frac{v(x\wedge_{\mathsf{L}} u)}{v(x)}$ to be an inclusion measure in lattice L.

∇

The proof of theorem 6 is given in the Appendix.

Note that $s(x,u)$ can be employed for indicating the degree of inclusion of x in u, therefore in the sequel $s(x,u)$ will also be denoted by $s(x\leq_{\mathsf{L}} u)$. Furthermore $s(x\leq_{\mathsf{L}} u)$ equals 1 if and only if $x\leq_{\mathsf{L}} u$. It might be useful to point out that an employment of function $f(x,u)=\frac{v(x\wedge_{\mathsf{L}} u)}{v(u)}$, with $v(u)$ in the denominator instead of $v(x)$, for indicating the degree of inclusion of x in u is "counter-intuitive" for the following reason. Let there be x,u in L such that $x\leq_{\mathsf{L}} u$ and $x\neq u$, that is let there be $x<_{\mathsf{L}} u$. Then an employment of function $f(x,u)$ implies that the degree of inclusion of x in u is less than 1, that is counter-intuitive. On the other hand, an employment of function $s(x,u)$ suggested by theorem 6 implies $x<_{\mathsf{L}} u \Rightarrow s(x,u)=1$ as expected intuitively. The effectiveness of function $s(x,u)$ has also been confirmed experimentally as shown in section 5.

All the analysis in this section has been carried out with regards to a lattice L. However, lattice L could itself be a product lattice of N other lattices $\mathsf{L}_1,\ldots,\mathsf{L}_N$, namely *constituent lattices*, that is $\mathsf{L}=\mathsf{L}_1\times\ldots\times\mathsf{L}_N$. The lattice ordering of the product of N lattices $\mathsf{L}_1,\ldots,\mathsf{L}_N$ accounts for the *modular* and *inherently hierarchic* nature of the *FL-framework* as it has been illustrated in [Petridis99]. The following result has been copied from [Petridis99].

Proposition 7 If $v_1,\ldots,v_N$ are valuations on lattices $\mathsf{L}_1,\ldots,\mathsf{L}_N$, respectively, then function $v=v_1+\ldots+v_N$ is a valuation on the product lattice $\mathsf{L}=\mathsf{L}_1\times\ldots\times\mathsf{L}_N$.

∇

Note that it suffices all valuations $v_1,\ldots,v_N$ to be *monotone* so as valuation $v=v_1+\ldots+v_N$ to be monotone as well. If at least one of the monotone valuations $v_1,\ldots,v_N$ is, in addition, a *positive valuation* then v is a positive valuation in product lattice $\mathsf{L}=\mathsf{L}_1\times\ldots\times\mathsf{L}_N$. The latter accounts for *FL-framework*'s capacity to treat jointly and with mathematical rigor disparate types of data. Note that the capacity for learning based on disparate types of data in the *FL-framework* has already been demonstrated in various

constituent lattices including: the lattice "unit hypercube" [Kaburlasos97], a lattice of fuzzy sets [Petridis98], and a lattice of symbols [Petridis99].

Various types of "data processing" are also possible in the *FL-framework*. For instance, *F*uzzy *L*attice *N*eurocomputing or *FLN* for short, is a possibility as shown in this paper. Another possibility for data processing in the *FL-framework* is regression as it will be shown elsewhere. The following section studies a specific lattice, that is the lattice of directed graphs, in view of an application involving graphs.

3 A Lattice of Directed Graphs

In this section a lattice G of directed graphs is shown followed by the definition of a positive valuation function in G. An example illustrates calculation of the degree of inclusion of a graph into another one in lattice G. The notion of a *graph* is defined formally underneath.

Definition 8 A **graph** G, denoted by $\mathsf{G}=(\mathsf{V},\mathsf{E})$, is the set-union of a finite non-empty set $\mathsf{V}=\{n_1,\ldots,n_N\}$ of **vertices** (or, **nodes**) and a set E of **edges** (or, **links**), where an edge is an ordered pair (n_i,n_j) of nodes n_i,n_j in V.

∇

The non-empty set $\mathsf{V}=\{n_1,\ldots,n_N\}$ contains the labels of a graph's nodes, whereas set E includes ordered pairs (n_i,n_j), $n_i,n_j\in\mathsf{V}$, $i,j\in\{1,\ldots,N\}$. It is important to note that E is a binary relation on V in the sense that a link (n_i,n_j) is eligible to appear in E if and only if both n_i and n_j are in the corresponding non-empty set V of nodes. Definition 8 implies that a graph might have no links, that is a graph $\mathsf{G}=(\mathsf{V},\mathsf{E})$ with $\mathsf{E}=\varnothing$ is eligible. Nevertheless for a link (n_i,n_j) in E both n_i and n_j have to be in V, hence a graph $\mathsf{G}=(\mathsf{V},\mathsf{E})$ with $\mathsf{V}=\varnothing$ and $\mathsf{E}\neq\varnothing$ is not allowed.

All graphs considered in this work are *directed graphs* in the sense that link (n_j,n_i) with $n_j\neq n_i$ is different than link (n_i,n_j). From henceforward a *directed graph* will also be called *graph* for simplicity. Note that a (directed) graph $\mathsf{G}=(\mathsf{V},\mathsf{E})$ may also be denoted, alternatively, as the set-union of V and E, that is $\mathsf{G}=\mathsf{V}\cup\mathsf{E}$, under the aforementioned condition that E is a binary relation on V.

The largest directed graph of interest in a particular application is the *universe of discourse*, namely *master-graph*, and it will be denoted by $\mathsf{M}=(\mathsf{V}_M,\mathsf{E}_M)$. Any particular graph $\mathsf{G}=(\mathsf{V},\mathsf{E})$, which might arise in the application in question, will be included in the master-graph, symbolically $\mathsf{G}\subseteq\mathsf{M}$. The latter, set-theoretic relation $\mathsf{G}=(\mathsf{V},\mathsf{E})\subseteq(\mathsf{V}_M,\mathsf{E}_M)=\mathsf{M}$ is interpreted as conjunction "$\mathsf{V}\subseteq\mathsf{V}_M$".AND."$\mathsf{E}\subseteq\mathsf{E}_M$".

The previous discussion implies that a graph G is in the power set $\wp(\mathsf{M})$ of the master-graph $\mathsf{M}=\mathsf{V}_\mathsf{M}\cup\mathsf{E}_\mathsf{M}$. Note, however, that not any element in the power set $\wp(\mathsf{M})$ of master-graph $\mathsf{M}=\mathsf{V}_\mathsf{M}\cup\mathsf{E}_\mathsf{M}$ is a graph. For instance, consider a non-empty set $\mathsf{S}\neq\varnothing$ which includes only links from E_M but no nodes from V_M. In line with definition 8, it follows that S is not eligible for representing a graph because S does not include any nodes. In conclusion, if G denotes the set of (directed) graphs emanating from a master-graph M it follows $\mathsf{G}\subset\wp(\mathsf{M})$, that is the set G of graphs is a proper subset of the power set $\wp(\mathsf{M})$ of the master-graph M.

It is well known that the power set $\wp(\mathsf{M})$ is lattice-ordered and moreover it is known that the corresponding lattice-ordering ($\leq_\wp$), lattice-meet ($\wedge_\wp$), and lattice-join ($\vee_\wp$) in $\wp$ are, respectively, the conventional set-inclusion ($\subseteq$), set-intersection ($\cap$), and set-union ($\cup$). It is quite straight forward to show that the aforementioned set G of graphs is *closed* under both operations set-union ($\cup$) and set-intersection ($\cap$), in the sense that if both G_1 and G_2 are in G then it follows that both $\mathsf{G}_1\cup\mathsf{G}_2$ and $\mathsf{G}_1\cap\mathsf{G}_2$ are in G. In conclusion, the set G of graphs emanating from a master-graph M is a lattice. As a convention the join ($\vee_\mathsf{G}$) and meet ($\wedge_\mathsf{G}$) operators in lattice G will be denoted, alternatively, by the set-union ($\cup$) and set-intersection ($\cap$) operators, respectively. Likewise, inclusion ($\leq_\mathsf{G}$) in lattice G will be denoted, alternatively, by set-inclusion ($\subseteq$). Lattice G is a complete lattice; the least element O_G and the greatest element I_G in G are the empty set and the master-graph M, respectively.

A positive valuation function can be defined in lattice G of directed graphs by a real function v: $\mathsf{G}\rightarrow\mathsf{R}$ which assigns a positive real number to a graph's node as well as to a graph's link. It can be shown easily that such a function v satisfies both $v(\mathsf{G}_1)+v(\mathsf{G}_2)=v(\mathsf{G}_1\vee_\mathsf{G}\mathsf{G}_2)+v(\mathsf{G}_1\wedge_\mathsf{G}\mathsf{G}_2)$ and $\mathsf{G}_1<_\mathsf{G}\mathsf{G}_2\Rightarrow v(\mathsf{G}_1)<v(\mathsf{G}_1)$ for two graphs $\mathsf{G}_1,\mathsf{G}_2$ in G, hence function v is a positive valuation function. The positive real number assigned to a graph's node n will be called *node weight*, denoted by w_n. Likewise, the positive real number assigned to a graph's link l will be called *link weight*, denoted by w_l. Apparently, "uncountably infinite many" positive valuations functions can be defined in G. In the interest of simplicity this work employs only positive valuation functions such that the weight of either a node or a link is equal to 1, that is $w_n= w_l=1$ for all nodes and links in master-graph M. The following example illustrates an employment of lattice inclusion measure s, defined by theorem 6, in the lattice G of graphs.

Example #1:

Consider the graphs G_1, G_2, and G shown in Fig.1. Graph $G_1=(V_1,E_1)$, is specified by $V_1=\{2,3,6\}$ and $E_1=\{(2,6),(6,2),(2,3),(3,2),(3,6)\}$. Note that none of the links (1,2), (1,3), (6,8), and (7,6) are in E_1 because a node of the latter links is not in V_1. In conclusion, graph G_1 may be denoted by the set-union of sets V_1 and E_1, that is $G_1=\{2,3,6,(2,6),(6,2),(2,3),(3,2),(3,6)\}$. The cardinality of the sets V_1 and E_1 is, respectively, $|V_1|=3$, $|E_1|=5$.

Likewise, graphs G_2 and G are defined, respectively, by $G_2=\{4,5,7,8,(4,5),(5,8),(8,7),(7,4)\}$ and $G=\{1,2,3,4,5,(2,3),(3,2),(1,2),(1,3),(1,1),(1,5),(4,5)\}$. In the sequel it is demonstrated the calculation of the degree of inclusion of graph G in graphs G_1 and G_2, respectively, using inclusion measure *s* defined by theorem 6. Note that

$G \wedge_G G_1 = G \cap G_1 = \{2,3,(2,3),(3,2)\}$, and

$G \wedge_G G_2 = G \cap G_2 = \{4,5,(4,5)\}$

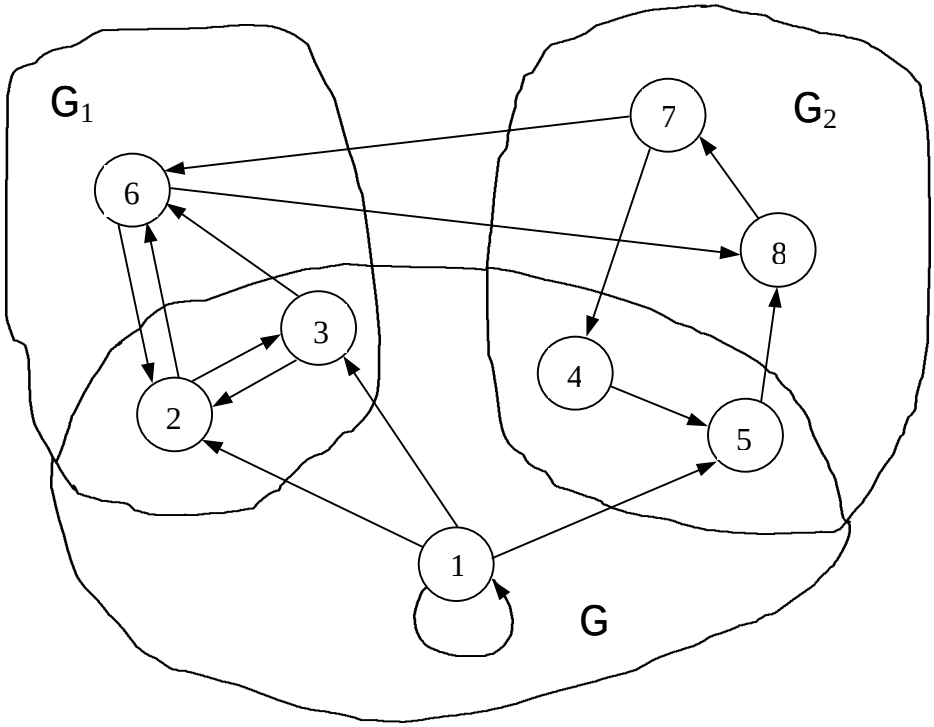


Figure 1

Graph G is included, in a lattice inclusion measure *s*(.) sense, more in graph G_1 than it is in graph G_2.

Let w_n denote the weight of a node and let w_l denote the weight of a link. Then

$$s(\mathsf{G} \leq_{\mathsf{G}} \mathsf{G}_1) = \frac{v(\mathsf{G} \wedge_{\mathsf{G}} \mathsf{G}_1)}{v(\mathsf{G})} \Rightarrow s(\mathsf{G} \subseteq \mathsf{G}_1) = \frac{v(\mathsf{G} \cap \mathsf{G}_1)}{v(\mathsf{G})} = \frac{2w_n + 2w_l}{5w_n + 7w_l}, \text{ and}$$

$$s(\mathsf{G} \leq_{\mathsf{G}} \mathsf{G}_2) = \frac{v(\mathsf{G} \wedge_{\mathsf{G}} \mathsf{G}_2)}{v(\mathsf{G})} \Rightarrow s(\mathsf{G} \subseteq \mathsf{G}_2) = \frac{v(\mathsf{G} \cap \mathsf{G}_2)}{v(\mathsf{G})} = \frac{2w_n + w_l}{5w_n + 7w_l}.$$

Using $w_n = w_l = 1.0$ it follows $s(\mathsf{G} \subseteq \mathsf{G}_1) = 0.34 > 0.25 = s(\mathsf{G} \subseteq \mathsf{G}_2)$, that is graph G is included more in graph G_1 than in graph G_2. Note that inclusion measure $s(\mathsf{G} \subseteq \mathsf{G}_1)$ quantifies, in principle according to definition 4, by a number in the range [0,1], the degree of inclusion of graph G in graph G_1. In other words, the total number of nodes and links in either G or G_1 does not "show" in number $s(\mathsf{G} \subseteq \mathsf{G}_1)$; the latter number only indicates "by how much graph G is inside graph G_1".

∇

4 σ- Fuzzy Lattice Neurocomputing (*s-FLN*) Scheme for Graph Clustering

The *s- F*uzzy *L*attice *N*eurocomputing (*s-FLN*) scheme is presented in this work for application to lattice G of directed graphs defined in the previous section. Note that the *s-FLN* scheme has been introduced under the name *FLNN* in [Petridis98], where its capacity for pattern recognition has been demonstrated on vector-based data sets as well as on synthetic data in a lattice of fuzzy sets. Moreover, the *s-FLN* scheme has been presented under the name *s-FLL* in [Petridis99], where it has been applied as well to a benchmark data set including a constituent lattice of symbolic data. Lately the authors have decided to switch to the name "*s-FLN* scheme" so as to adhere to the terminology established subsequently in the *FL-framework*. In particular, the initials "*FL*" denote a scheme applicable in the *FL-framework*, initial "*N*" signifies a neural implementation, and letter *s* indicates that both the training phase and the testing phase are carried out using an inclusion measure (s).

The *s-FLN* has been inspired from the biologically motivated *A*daptive *R*esonance *T*heory (ART) neural paradigm [Carpenter87], in particular from fuzzy-ART [Carpenter91]. It has been explained in [Petridis98] that both training and testing by *s-FLN* are effected the same way as by fuzzy-ART. Moreover, well-known properties of learning by the fuzzy-ART neural model [Huang], [Georgiopoulos] are retained by *s-FLN*.

In order to better illustrate the mechanics involved in learning by *s-FLN* a conventional neural architecture is reviewed in Fig.2. In particular, Fig.2(a) shows two successive layers including the *Input Layer* of a conventional fully interconnected neural architecture. Note that in Fig.2(a) an interconnection

between two neurons corresponds to a single real number. Moreover, the outputs of *Upper Layer* are fed upwards to another layer of neurons which is not shown in Fig.2(a). Fig.2(b) resumes concisely the neural architecture of Fig.2(a). More specifically, the *Input Layer* in Fig.2(b) is shown as a single neural node. Hence, a line interconnection between the *Input Layer* neuron of Fig.2(b) and an *Upper Layer* neuron corresponds to a vector of numbers. That is, the neural architecture in Fig.2(b) underlines that a datum dealt with in Fig.2(b) is a vector. Pursuing further Fig.2(b) note that a neuron in *Upper Layer* computes the sigmoid function of the inner product of two vectors, the latter are 1) input vector x, and 2) a weight vector w_i, i=1,…,L. Recall that the *sigmoid* is a monotonically increasing function of its argument, where the latter 'argument' is in particular the aforementioned inner product of two vectors. It can be argued that the outcome of the inner product operation may quantify by a number in interval [0,1] the similarity of its operands. It is shown in the sequel that the *s-FLN* carries out a "like" data processing procedure.

The corresponding neural architecture of *s-FLN* for processing graphs is shown in Fig.3. The names and roles of the various subsystems in Fig.3 are analogous to the names and roles of the corresponding subsystems in an *A*daptive *R*esonance *T*heory (ART) neural model as explained in [Petridis98] and they will not be repeated again. It is worthwhile noting the structural similarity of Fig.3 to Fig.2(b). A key difference between Fig.2(b) and Fig.3 is that the former processes vectors of numbers whereas the latter processes graphs. That is, a datum dealt with in Fig.2(b) is a vector of numbers, whereas a datum dealt with in Fig.3 is a graph represented as a set of nodes and links in line with definition 8. Likewise note that the weight G_k, k=1,…,L of an interconnection in the *s-FLN* architecture in Fig.3 corresponds to a whole graph. Finally, a neuron in *Category Layer* F_2 of the *s-FLN* architecture (Fig.3) computes the degree of inclusion of an input graph R_i to a weight graph G_k, k=1,…,L using inclusion measure function $s(R_i \leq_G G_k)=s(R_i \subseteq G_k)$, the latter is a number in interval [0,1].

We remark that neural computing based on the elements of a mathematical lattice, including the totally ordered lattice of real numbers, is *FL-framework*'s "proposal" in order to enhance conventional neural computing which is typically carried out on real numbers. It should also be pointed out *s-FLN*'s capacity to process, as well, elements of a product lattice $L=L_1\times\ldots\times L_N$ involving disparate constituent lattices $L_1,\ldots,L_N$. The latter capacity accounts for *s-FLN*'s potential for dealing jointly with disparate types of data [Petridis99]. Nevertheless, in the context of this work, a single constituent lattice is considered, that is in particular the lattice G of directed graphs emanating from a master-graph. Algorithms for training- and for testing- using the *s-FLN* are described in the following in pseudo code form.

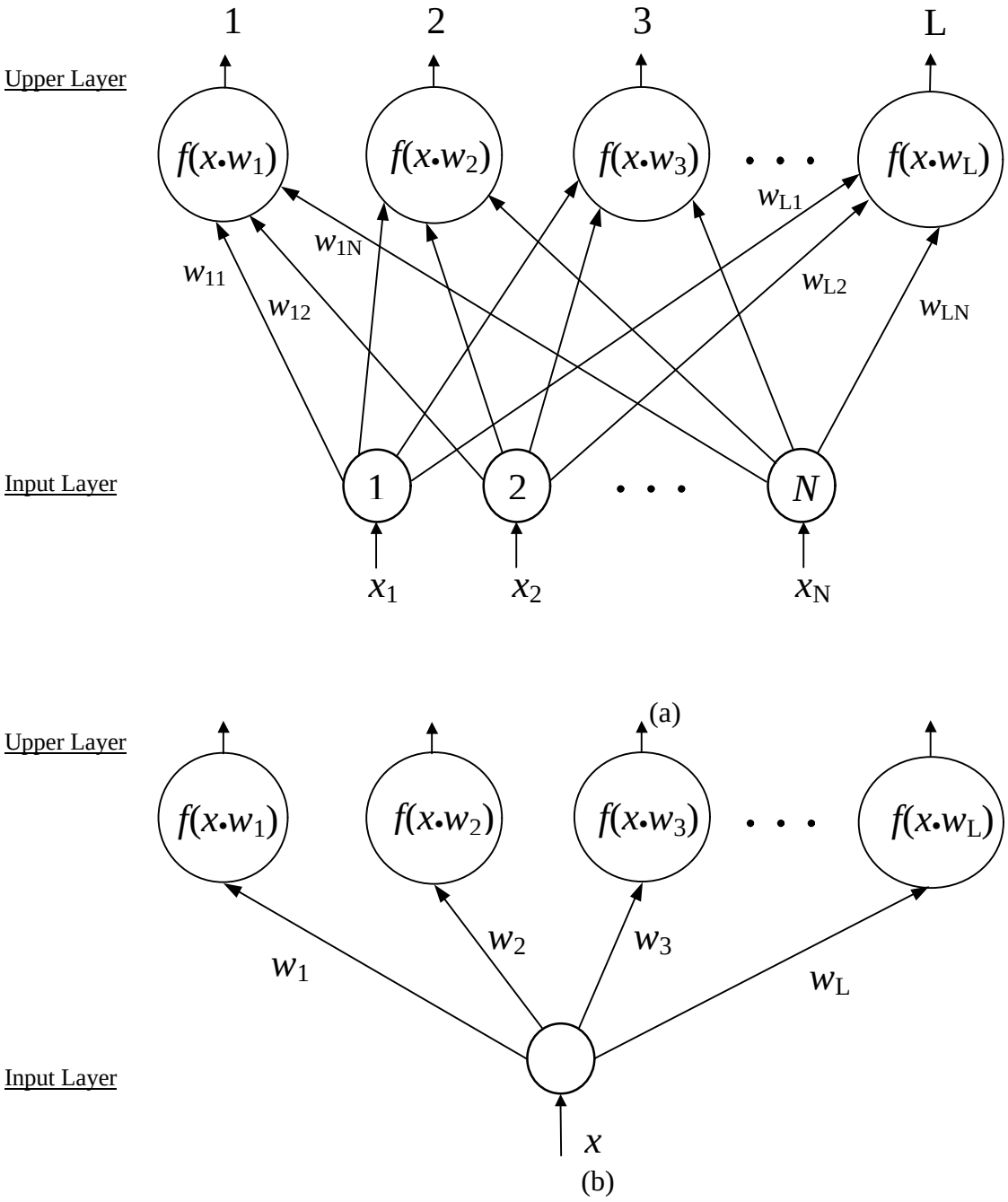
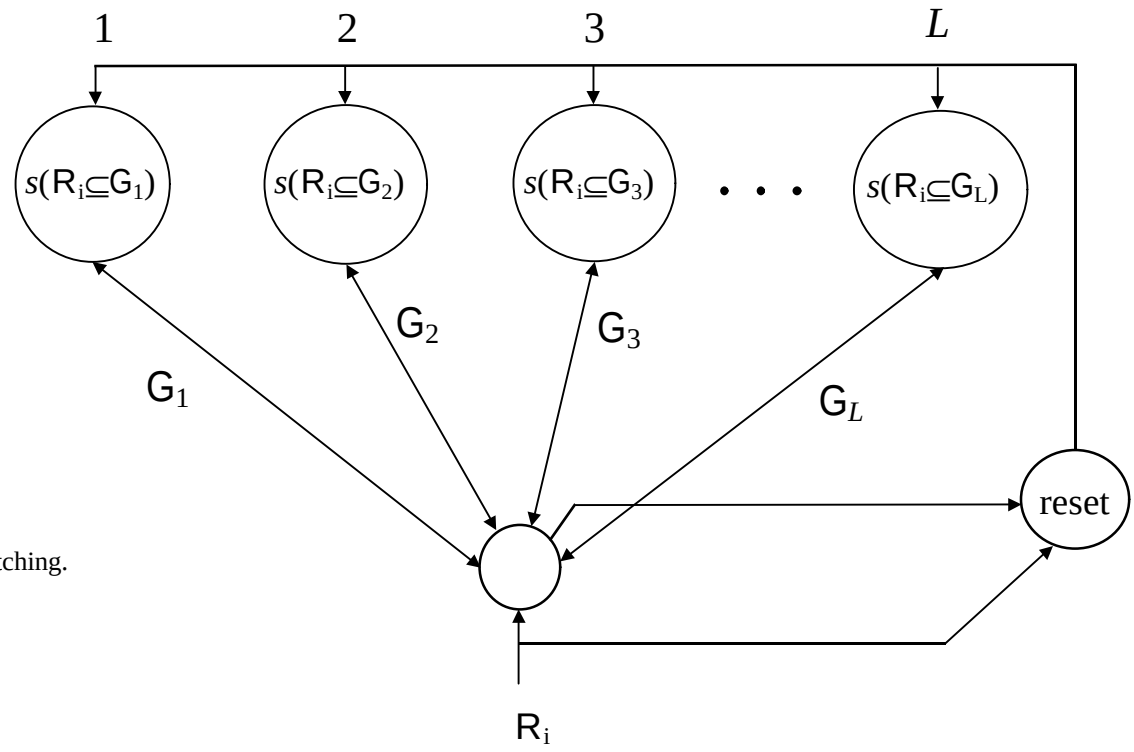


Figure 2

(a) Two successive layers of a conventional fully connected neural architecture including the *Input Layer*. Due to the dot-product between the outputs of *Input Layer* with the connection weights to *Upper Layer* the input to an upper layer neuron may quantify by a number in interval [0,1] the "similarity" of input vector $x=(x_1,x_2,\ldots,x_N)$ to the stored weights $w_1=(w_{11},w_{12},\ldots,w_{1N}),\ldots, w_L=(w_{L1},w_{L2},\ldots,w_{LN})$.

(b) A concise representation of two successive layers of a conventional fully connected neural architecture. The *Input Layer* is shown as a single neural node, which can accommodate a vector of numbers. An interconnection between *Input Layer* and *Upper Layer* accommodates a vector of numbers.

Category Layer F_2
Competition : winner takes all.



Input Layer F_1
Buffering & Matching.

Figure 3

The two layer *s- F*uzzy *L*attice *N*eurocomputing (*s-FLN*) architecture for processing directed graphs. *L*, is the number of *Category Layer* neurons, which equals the total number of clusters/graphs. A *Category Layer* neuron employs a lattice inclusion measure *s*(.) as its activation function in order to specify by a number in interval [0,1] the degree of inclusion of an input graph R_i to a weight graph G_k, k=1,…,L. *Input Layer* is used to buffer an input graph. A "reset" node is used for resetting the activity of a node in the *Category Layer*.

Algorithm for Training by *s-FLN*

Function trainFLN(S,ρ)

1 L:=1;
2 $G_L = R_0$, (the first input graph $R_0 \in G$ is memorized)
3 **For** each graph R_i, i=1,…,n in the training set S **do**
4 "Set" all memorized-graphs G_k, k=1,…,L
5 Calculate $s(R_i \leq_G G_k)$, k=1,…,L
6 **while** (there exist memorized-graphs G_k, k=1,…,L "set") **do**
7 G_J:= the graph with $max\{s(R_i \leq_G G_k), k=1,\ldots,L\}$ among the "set" memorized-graphs
8 **if** $(s(G_J \leq_G R_i) \geq \rho)$ **then** $G_J := G_J \vee_G R_i$; exit the **while** loop;
9 **else** "reset" G_J;
10 **endwhile**
11 **if** (all memorized-graphs G_k, k=1,…,L have been "reset") **then**
12 L:=L+1;
13 $G_L := R_i$; (memorize input graph $R_i \in G$)
14 **endif**
15 **endfor**
16 **return** G_k, k=1,…,L

Various terms employed by the *s-FLN* algorithm above including "set", "reset", "vigilance parameter" originate from ART as explained in [Petridis98]. More specifically, a node in Category Layer (Fig.3) is "set" when the node in question is potentially available for accommodating an input to the *s-FLN*, whereas a "reset" node is unavailable for doing so. Furthermore, the "vigilance parameter" ρ is a user-defined threshold parameter in the range [0,1] such that when a particular degree of lattice inclusion is above the vigilance threshold then a learning procedure may be triggered. The principal difference between fuzzy-ART and *s-FLN* concerns the type of data dealt with the two neural models. In particular, *s-FLN* deals with intervals of data in a lattice domain including fuzzy-ART's domain, the latter is the unit N-dimensional hypercube in which fuzzy-ART deals solely with points, these are trivial intervals in the N-dimensional Euclidean space. More details regarding a comparison of *s-FLN* with fuzzy-ART are shown in [Kaburlasos00].

Note that function trainFLN(S,ρ) above requires two input arguments, these are a set S which includes all graphs to be used for training as well as the "vigilance parameter" ρ, that is a real number between 0 and 1. The output of function trainFLN(S,ρ) is a set of clusters, these are graphs G_k, k=1,…,L which have been learned during training. In the previous scheme the learning is continuously "on" for all inputs. In particular, a new input graph $R_i \in G$ is assigned to the most activated memorized-graph G_J, $J \in \{1,\ldots,L\}$ provided that $\sigma(G_J \leq_G R_i)$ is larger than the user-defined "vigilance parameter" ρ. Otherwise "reset" is triggered and the quest for a new winner resumes. If all graphs $G_1,\ldots,G_L$ have been "reset" and no winner has been found then input graph R_i is memorized as indicated in line 13 of the above algorithm. Memorization means that input graph R_i is stored as a new cluster/graph in the *Category Layer* (Fig.3).

Learning by *s-FLN* is effected in steps 8 and 13 of the previous algorithm. On the one hand, learning in step 13 has already been explained. On the other hand, learning in step 8 is effected by the lattice G join operator $\vee_G$, that is the set-inclusion operation $\cup$, as explained in section 3. Note that a memorized-graph G_k, k=1,…,L can be regarded as interval $[\varnothing, G_k]$, where ∅ is the empty set the latter is the least element O_G in the complete lattice G of directed graphs. Hence, a graph G belongs in interval $[\varnothing, G_k] = [O_G, G_k]$ if and only if $\varnothing = O_G \subseteq G \subseteq G_k$. In conclusion, it follows that learning by *s-FLN* is effected either by memorizing intervals of graphs in G (step 13) or by enhancing existing intervals of graphs in G (step 8) using the lattice G join operator $\vee_G$.

To employ the *s-FLN* solely for testing, *s-FLN*'s "learning capacity" must be disengaged. In particular testing is affected according to the following algorithm. Note that function testFLN(T) underneath requires one input argument that is set T which includes all graphs to be used for testing.

Algorithm for Testing by *s-FLN*

Function testFLN(T)

1 **For** each graph R_m, m=1,…,N in the testing set T **do**

2 Calculate $s(R_m \leq_G G_k)$, k=1,…,L

3 Assign R_m to G_J:= the graph with $max\{s(R_m \leq_G G_k), k=1,\ldots,L\}$

4 **endfor**

Note that the number L of graphs/clusters in *Category Layer* F_2 remains constant during testing.

In all, the *s-FLN* scheme for clustering is a competitive self-organizing neural scheme, which employs a lattice inclusion measure σ aiming at crisp set identification by clustering. The capacity of *s-FLN* to generalize on new and hitherto unknown patterns is due to the employment of inclusion measure σ, and it has been demonstrated on disparate lattice domains including vectors of real numbers [Kaburlasos97], fuzzy sets [Petridis98], and lattice-ordered symbols [Petridis99]. The total number L of graphs/clusters to be learned is not known *a priori*, more specifically number L is specified 'on-line' during training. Regarding the training complexity, it follows from function trainFLN(S,ρ) that learning by *s-FLN* is achieved in one pass through the training data set, that is there are no multiple cycles in the algorithm. The worse case training scenario would be to keep "resetting" all L clusters for every input. Hence the training complexity is quadratic $O(n^2)$, where n is the number of graphs for training.

The same way as with an ART neural model [Carpenter87], [Carpenter91], retraining the *s-FLN* by another data set does not "wash away" previous learning. More specifically, retraining the *s-FLN* by a new data set either enhances previously learned clusters (step 8) or it creates new clusters (step 13) in the *Category Layer*. The "vigilance parameter" ρ regulates *s-FLN*'s *granularity of learning*, that is the number of clusters/graphs in the *Category Layer*. More specifically, on the one hand, a large value of ρ within interval [0,1] implies more clusters/graphs in the *Category Layer* F_2 and hence a "more refined" knowledge is attained, on the other hand, as ρ decreases, fewer clusters are learned. Note that the aforementioned role of the "vigilance parameter" ρ as a "regulator for learning" has been retained from an *A*daptive *R*esonance *T*heory (ART) neural model [Carpenter91].

5 Graph Clustering for Semantic Document Classification

5.1 A Domain for Graph Clustering

The proposed methods have been applied for synonym clustering in a master-graph stemming as described in the following from MOBY Thesaurus of synonyms, which is available from The Institute for Language, Speech and Hearing, University of Sheffield [MOBY]. MOBY Thesaurus contains in an ASCII file records with terms such that the first term in a record, namely *root term*, is followed by a list of synonym terms in alphabetical order. Each term, including the root, is followed by a comma. The last term in a record is followed by a carriage return /linefeed. There are 30,260 records in MOBY Thesaurus and more than 2.5 million synonym terms, in all.

In a series of preprocessing steps, MOBY Thesaurus has been reduced in size by omitting uncommon terms as described in the following. In the first place, all synonym terms which do not appear as root terms have been eliminated. Hence a simplified Thesaurus emerged, namely *simplified MOBY Thesaurus*, including 681,761 synonym terms. Further preprocessing has been carried by removing root terms including two or more words, for example such root terms as "a capella", "bid price", "infantile paralysis", etc. have been removed. Finally, *rare* root terms have also been removed, where a root term is considered to be *rare* if the number of times it occurs as a synonym in MOBY Thesaurus is below a user-defined threshold. In conclusion a new Thesaurus emerged, namely *reduced MOBY Thesaurus*, including 11,769 root terms and 628,242 synonyms terms, in all. Note that a *root term* does not always include a single word; for instance root terms consisting of hyphen-separated words are also included, such as "well-versed", "self-esteem", "well-to-do", etc. The number of synonyms in a root term is called *cardinality* of the corresponding root term.

A master-graph M with 11,769 nodes, or vertices, has emerged as follows. Each root term of the reduced MOBY Thesaurus corresponded to a node in the master-graph. The existence of a link (n_i,n_j) in M from a node n_i to another node n_j was assumed if and only if the term corresponding to node n_j is in the list of synonyms of the root term corresponding to node n_i. In this way a master-graph with 11,769 nodes and 628,242 links emerged from the reduced MOBY Thesaurus.

The goal of clustering by *s-FLN* in the master-graph M has been to learn clusters of terms that retain a similar meaning. The term *hyperword* has been coined to denote such a cluster of terms. The term *cardinality of a hyperword* denotes the number of terms clustered in a hyperword. Note that the *granularity of learning* defined at the end of section 4 equals, in this case, the total number of computed hyperwords. In the context of this work hyperwords are meant for dimensionality reduction in a document classification problem by replacing a term in a document by its corresponding hyperword. Note that, often, in a document classification problem every word in the document is used as a feature. Furthermore, various dimensionality reduction techniques have been considered and employed in the literature based on document frequency thresholding, information gain, mutual information, term strength, and χ^2 [Drucker], [Quinlan]. In addition, "word stemming" and a "stop list" have also been used in order to reduce the dimensionality of the feature space [Mladenic98], [Sahami]. This work proposes a novel method for reducing dimensionality of the feature space for document classification based on semantics.

5.2 *s-FLN* for Clustering Words in a Thesaurus

A specific type of graphs, namely *s-Graphs*, have been used in the experiments here. In particular, the set V of nodes in an s-Graph $G=(V,E)$ is partitioned as it is explained underneath to two disjoint sets

V_c and V_s, namely *set of core nodes* and *set of satellite nodes*, respectively, such that $V=V_c\cup V_s$. Likewise, the set E of links is partitioned to two disjoint sets E_c and E_s, namely *set of core links* and *set of satellite links*, respectively, such that $E=E_c\cup E_s$. The set E_c of core links contains all the links in master-graph M interconnecting core nodes in V_c and only those links, whereas the set E_s of satellite links contains all the links in M from a core node to another node in M, the latter nodes are called *satellite nodes*. Recall that the set of satellite nodes has been denoted above by V_s. Hence, by construction of an s-Graph, a satellite node corresponds to a synonym of a core node. Note that when a node appears both as a core- and as a satellite- node then the node in question counts as a core node in an s-Graph. The previous s-Graphs are *1st-order s-Graphs*, in the sense that the maximum number of links between a core node and a satellite node is one. An *nth-order s-Graph* is an s-Graph such that the aforementioned maximum number of links between a core node and a satellite node equals *n*. Only 1st-order s-Graphs have been considered in this work and the graphs in question are called *graphs* for simplicity. Note that a *singleton graph* is a graph with a single core node only.

Clustering by *s-FLN* in the reduced MOBY Thesaurus has been effected by feeding the *s-FLN* with a series of its 11,769 root terms. Each root term has been presented as a singleton graph with as many satellite nodes as the synonyms of the root term in question are. The algorithm described in section 4 by function trainFLN(S,ρ) has been applied in principle during the experiments reported in the sequel. Nevertheless, in order to produce hyperwords which contain semantically related terms the following conjunctive condition has been considered, in addition, in line 7 of function trainFLN(S,ρ): "($s(R_i\leq_G C_j)\geq\rho_1$) for all singleton graphs $C_j\subseteq G_J$". AND. "($s(R_i\leq_G C_j)\geq\rho_2$) for at least one singleton graph $C_j\subseteq G_J$", where both ρ_1 and ρ_2 are in the range [0,1]. The reason for employing the aforementioned conjunction is to keep a minimal semantic affinity of a new term to be incorporated in a hyperword to all other terms in the hyperword in question. Note also that the aforementioned conjunction can be easily accommodated in hardware in *s-FLN* architecture's "reset subsystem" (Fig.3), by employing two comparators and an AND gate.

It should be pointed out that in this particular application of *s-FLN* for graph clustering, inclusion measure $s(x,u)=\dfrac{v(x\wedge_L u)}{v(x)}$ has given better results than inclusion measure $k(x,u)=\dfrac{v(u)}{v(x\vee_L u)}$ because the hyperwords computed using $s(x,u)$ include only terms semantically related to each other, whereas it has been verified experimentally that the hyperwords produced using $k(x,u)$ may include terms which are semantically quite different from one another. The reason for the latter difference is that for two disjoint graphs x and u, $k(x,u)$ implies a non-zero degree of inclusion of x in u, whereas $s(x,u)$ implies a zero degree of inclusion of x in u and hence it implies a better decision-making in computing hyperwords.

A graph can be represented in the computer memory using a sparse square "adjacency matrix". Nevertheless, since the total number of nodes and links could vary substantially during learning by *s-FLN*, a graph has been represented in the experiments underneath by a linked list which implies dynamic memory allocation. Furthermore, due to the inherently hierarchic and modular nature of the *s-FLN* scheme, an object oriented programming language, in particular C++, has been employed for simulating the *s-FLN* scheme. Certain mechanisms of an object oriented programming language including 1) inheritance, and 2) operator overloading, have been particularly useful and convenient in implementing in software the inherently hierarchic and modular *s-FLN*.

5.3 Computation of Hyperwords

A series of experiments has been carried out for learning clusters of semantically related words, that is for learning *hyperwords*, by *s-FLN* with vigilance parameters ρ_1 and ρ_2 as explained in the previous section. It has been confirmed experimentally that the collection of hyperwords calculated by *s-FLN* depends on the order of presenting the 11,769 root terms of the reduced MOBY Thesaurus. Such a data order dependence is a well-known property of *s-FLN* [Petridis99]. Three different data orderings have been employed for clustering by the *s-FLN* with vigilance parameters ρ_1=0.001 and ρ_2=0.1, and the corresponding results are reported in Table 1. In particular, for data ordering DO1 the 11,769 root terms have been presented in alphabetical order. A total number of 7,833 hyperwords were calculated with cardinalities ranging from 1 to 213. Data ordering DO2 reported in Table 1 has been obtained by arranging the 11,769 root terms of the reduced MOBY Thesaurus in a decreasing order in their number of synonyms. That is, a root term with more synonyms has been presented before another root term with fewer synonyms, while two root terms with an equal number of synonyms have been presented at random. A total number of 8,899 hyperwords were calculated with cardinalities ranging from 1 to 232. Finally, data ordering DO3 has been obtained by reversing the order of data in DO2. That is, in DO3 a root term with fewer synonyms has been presented before another root term with more synonyms, resulting in a total number of 2,601 hyperwords with a number of core nodes ranging from 1 to 192. In conclusion, data ordering DO3 has resulted in the largest data compression from 11,769 root terms in the Thesaurus down to 2,601 hyperwords. The smaller granularity of learning implied by data ordering DO3 compared to the corresponding granularities implied by either DO1 or DO2 is attributed to the initial memorization by *s-FLN* in DO3 of graphs with a smaller number of synonyms. Hence, due to the conjunction which includes two vigilance parameters ρ_1 and ρ_2 described in the previous section, fewer hyperwords have been computed for data ordering DO3. Moreover, data ordering DO3 has resulted in hyperwords which include semantically similar terms as explained in the following.

TABLE 1
Three different orderings DO1, DO2, and DO3 of the records in the reduced MOBY thesaurus have resulted in, as explained in the text, different sets of hyperwords.

Data Ordering	Granularity of Learning	min cardinality of a hyperword	max cardinality of a hyperword	Processing Time [in min]
DO1	7,833	1	213	126
DO2	8,899	1	232	231
DO3	2,601	1	192	103

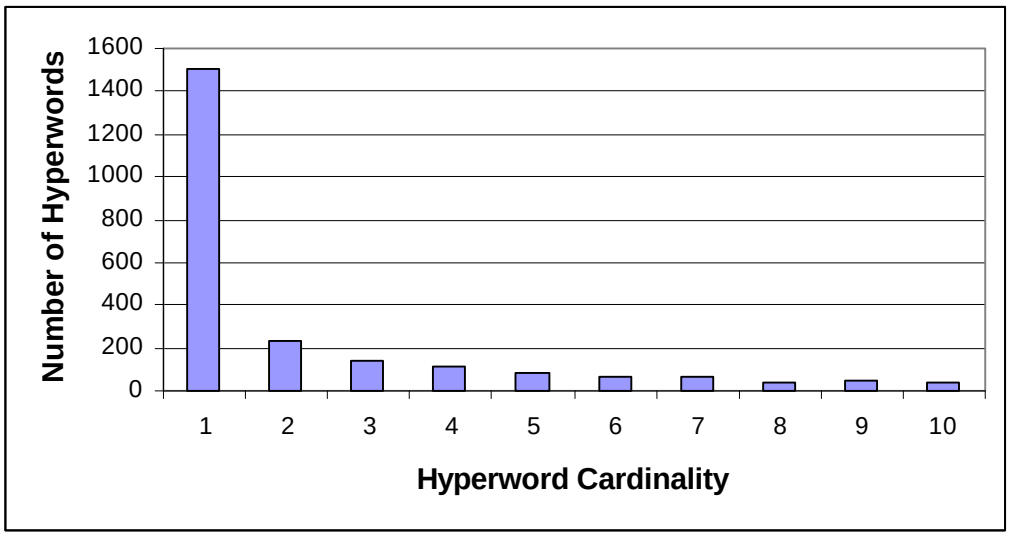
On the one hand, the hyperwords produced by either data ordering DO1 or data ordering DO2 have included several terms with different, even conflicting, semantics. On the other hand, data ordering DO3 has resulted in hyperwords including terms semantically similar to each other as shown in Table 2 which shows selected hyperword labels for hyperwords of various cardinalities together with the corresponding terms they cluster. It should also be noted that data ordering DO3 has resulted in a shorter data processing time as shown in Table 1. In particular, data ordering DO3 required 103 min, whereas data orderings DO1 and DO2 required 126 min and 231 min, respectively, on a Pentium III at 500 MHz with 128 RAM. Hence, in addition to providing with fewer hyperwords including terms which are semantically more similar to each other, data ordering DO3 is computationally the least expensive. Two figures with histograms are provided in the following in order to illustrate quantitatively the learning of hyperwords by *s-FLN* with data ordering DO3 and vigilance parameters ρ_1=0.001 and ρ_2=0.1.

TABLE 2

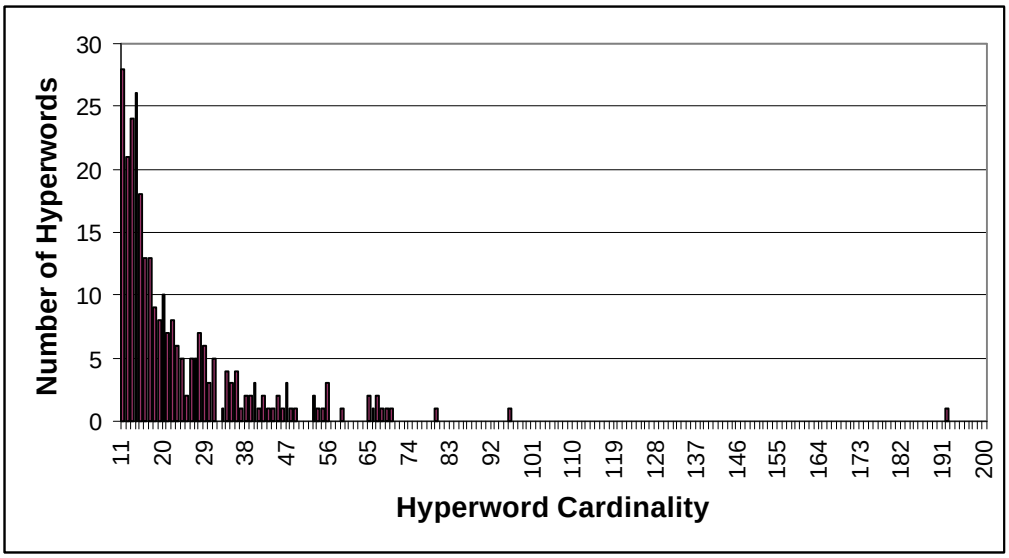
Hyperwords, these are clusters of semantically related words, have been calculated by "*s-FLN* for clustering" on the reduced MOBY thesaurus with parameters ρ_1=0.001, ρ_2=0.1. The words were fed to *s-FLN* in data ordering DO3 as explained in the text.

Hyperword ID	Root terms included in a hyperword
hw#24 (with 4 core nodes)	agriculture farming agrarian rural
hw#74 (with 6 core nodes)	housekeeper caretaker jailer keeper warden guardian
hw#138 (with 14 core nodes)	paranoid neurotic psychotic maniac demented maniacal touched deranged hysterical unbalanced lunatic distracted distraught mental
hw#157 (with 5 core nodes)	autocrat despot dictator oppressor tyrant
hw#269 (with 10 core nodes)	versus toward against across contra confronting inverse polar facing obverse
hw#401 (with 3 core nodes)	chauvinist dogmatist bigot
hw#405 (with 5 core nodes)	considerably greatly highly exceedingly awfully
hw#410 (with 30 core nodes)	dynamite decapitate disassemble unmake incinerate dismantle devastate vandalize vaporize pulverize devour demolish pillage slaughter shatter butcher batter atomize maul dissolve assault consume disintegrate overwhelm ravage spoil wreck kill confound destroy
hw#988 (with 29 core nodes)	amazing marvellous arresting wondrous memorable celebrated astonishing miraculous unusual esteemed puzzling uncommon phenomenal enigmatic wonderful remarkable noteworthy conspicuous salient unique noticeable marked notable singular outstanding special exceptional rare strange
hw#1059 (with 20 core nodes)	amazed captivated mesmerized spellbound charmed enthralled bewitched astounded surprised fascinated entranced aghast dumbfounded puzzled bewildered overwhelmed enchanted stupefied resigned confounded
hw#1079 (with 24 core nodes)	moratorium impasse obstacle congestion wait deterrent hindrance detention respite fixation closure impediment lag obstruction interruption restriction interference repression stoppage resistance suppression pause delay suspension
hw#1118 (with 67 core nodes)	corrupting damaging deleterious lethal pernicious detrimental baleful destructive hurtful poisonous injurious mischievous malevolent ominous harmful baneful distressing malignant noxious painful malign hateful ill dreadful villainous deadly deplorable fatal virulent killing miserable heinous sinister dire noisome grievous wretched wicked fell shocking terrible fierce vicious nasty black sad vile rotten grim offensive outrageous flagrant harsh atrocious dirty sore dark monstrous evil grave wrong bad severe gross rough foul heavy
hw#1253 (with 14 core nodes)	mirth enjoyment recreation fruition amusement glee euphoria well-being elation joy exhilaration happiness gaiety fun
hw#1562 (with 14 core nodes)	adulterated incomplete lacking damaged impaired blemished wanting unfinished partial imperfect deficient inadequate cracked defective
hw#1615 (with 10 core nodes)	inkling indication clue suspicion intimation symptom tinge tincture hint suggestion
hw#1765 (with 5 core nodes)	unbridled immoderate exorbitant excessive intemperate
hw#1883 (with 3 core nodes)	inexact inaccurate imprecise
hw#1925 (with 5 core nodes)	intention ambition inspiration motive aspiration
hw#2211 (with 4 core nodes)	infamous shameful notorious scandalous

On the one hand, Fig.4 shows the number of hyperwords versus hyperword cardinality. Note that there are 1502 hyperwords with cardinality "1" which account for approximately 57.74% of the total number of 2,601 hyperwords. Since the number of hyperwords decreases almost exponentially with hyperword cardinality, Fig.4 has been broken to Fig.4(a) for cardinalities 1 to 10, and Fig.4(b) for cardinalities 11 to 200.



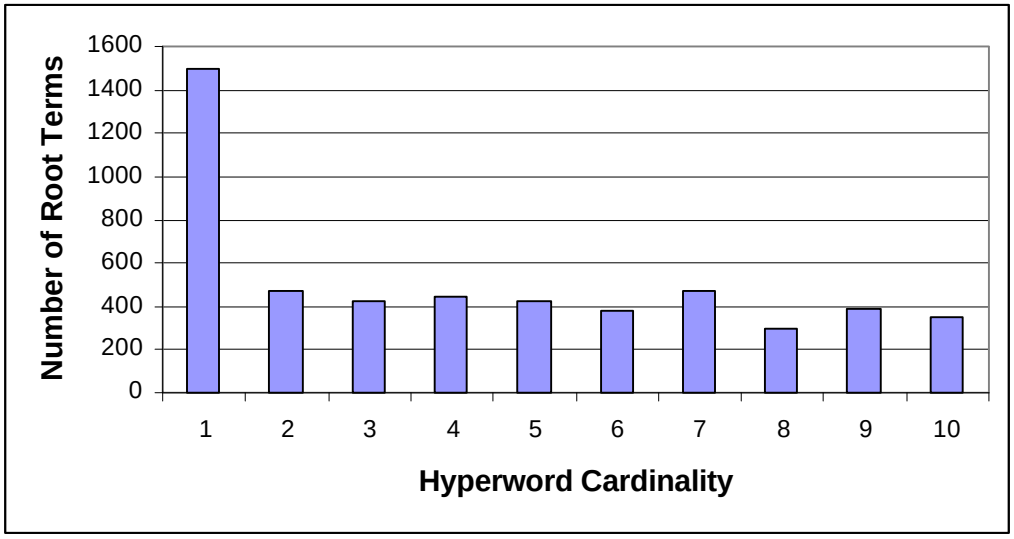
(a)



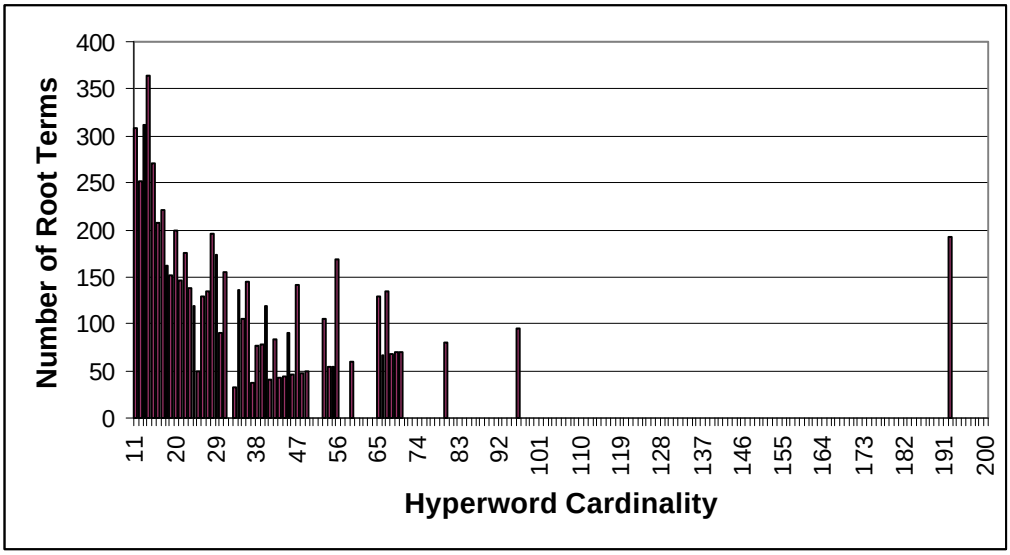
(b)

Figure 4
The histograms above show
(a) The number of hyperwords versus hyperword cardinality for cardinality sizes 1 to 10. Hyperwords with cardinality "1" account for approximately 57.74% of the total number of 2,601 hyperwords computed on the reduced MOBY Thesaurus.
(b) The number of hyperwords versus hyperword cardinality for cardinality sizes 11 to 200.

On the other hand, Fig.5 displays the number of root terms versus hyperword cardinality. Note that 1502 root terms, that is approximately 12.67% of the total number of 11,769 root terms in the reduced MOBY Thesaurus, correspond to hyperwords with cardinality "1". Likewise as in Fig.4, and for the same reason, Fig.5 is broken to two parts, these are Fig.5(a) and Fig.5(b). In both Fig.4(b) and Fig.5(b) it is shown that there is a single outlier hyperword with cardinality 192.



(a)



(b)

Figure 5
The histograms above show
(a) The number of root terms versus hyperword cardinality for cardinality sizes 1 to 10. Hyperwords with cardinality "1" represent approximately 12.76% of the total number of 11,769 root terms in the reduced MOBY Thesaurus.
(b) The number of root terms versus hyperword cardinality for cardinality sizes 11 to 200.

It is quite important to note that the vigilance parameters ρ_1 and ρ_2 can be used for fine-tuning the granularity of learning by *s-FLN* in a specific application. Various experiments have been carried out and the corresponding results are reported in the Table 3 for data ordering DO3. An increase in either ρ_1 or ρ_2 has implied an increase of the granularity of learning as shown in Table 3 due to the conjunction which involves the two vigilance parameters ρ_1 or ρ_2 as it has been described in the previous section.

TABLE 3
For data ordering DO3 various combinations of values of the vigilance parameters ρ_1 and ρ_2 have resulted in a different granularity of learning as explained in the text.

Vigilance parameter ρ_1	Vigilance parameter ρ_2	Granularity of Learning
0.001	0.1	2,601
0.001	0.2	2,792
0.001	0.3	3,638
0.001	0.4	5,486
0.01	0.1	2,802
0.05	0.1	3,694
0.1	0.1	4,329

A final remark concerns the definition of positive valuation function $v(.)$, that is the definition of weights w_n and w_l for a node and a link, respectively. Note that the significance of choosing a "good" positive valuation function in clustering by *s-FLN* has been reported for inclusion measure $k(x,u) = \dfrac{v(u)}{v(x \vee_L u)}$ in [Petridis99]. Nevertheless, it is expected theoretically that all positive valuation functions imply the same results of clustering when inclusion measure $s(x,u) = \dfrac{v(x \wedge_L u)}{v(x)}$ is employed. In particular given x, note that the ordering of numbers $\dfrac{v(x \wedge_L u_i)}{v(x)}$ does not change for different positive valuation functions hence it is expected that *s-FLN* always provides with identical clustering results for any underlying positive valuation function. The latter theoretically expected result has been verified experimentally, as well, using different positive valuation functions which have been defined for different values of a node weight (w_n) and a link weight (w_l).

5.4 Document Classification Based on Hyperwords

This section builds on the previous experimental results and empirical evidence is provided regarding the merit of hyperwords for document classification. For the experiments described in the following documents were drawn randomly from "Reuters-21578, Distribution 1.0" benchmark collection of documents [Lewis], and the following three document classification problems have been dealt with.

Problem-1:
A total number of 148 documents were drawn from the three categories: "colombia", "singapore", and "taiwan" from the *PLACES set of categories* of Reuters-21578. Approximately 2/3 of the total number of documents have been employed for training and the remaining 1/3 of the documents have been used for testing. The classification problem was to learn predicting the correct category of a document in the testing set.

Problem-2:
A total number of 143 documents were drawn from the two categories: "nasdaq", and "nyse" from the *EXCHANGES set of categories* of Reuters-21578. Approximately 2/3 and 1/3 of the total number of documents have been used, respectively, for training and testing.

Problem-3:
A total number of 169 documents were drawn from the two categories: "opec", and "worldbank" from the *ORGS set of categories* of Reuters-21578. Likewise as above, approximately 2/3 and 1/3 of the total number of documents have been used for training and testing, respectively.

Note that in each problem, the corresponding categories were represented by approximately equal numbers of documents. Moreover, note that only the text between tags <BODY> and </BODY> of a document has been used in this work for classification.

All documents have been preprocessed using a standard *stoplist* of words so as to remove trivial words such as letters, articles, etc. After preprocessing by a stoplist, the documents in problem-1 contained, respectively, an average of 71.79 words with standard deviation 51.59, whereas the documents in problems -2 and -3 contained, respectively, an average of 37.58 words with standard deviation 34.47, and an average of 95.77 words with standard deviation 51.11 as shown in Table 4. Preprocessing concluded by applying the Porter algorithm for stemming [Porter].

The *vocabulary of stemmed words* in a document classification problem included all different stemmed words, ordered alphabetically, which appear in all the documents of the problem in question. The sizes of the *vocabularies of stemmed words* have been 2482, 1585, and 3066 in problems 1, 2, and 3, respectively. In conclusion, a document has been represented by a vector of integers, where a vector-entry corresponded to a specific word in the *vocabulary of stemmed words*, and the (integer) value of a vector-entry indicated the number of occurrences of the stemmed word in question in a document.

In order to validate comparatively the effectiveness of hyperwords, out of each one of the preprocessed documents of stemmed words another document has been generated by replacing a stemmed word by its corresponding hyperword. Note that the "Porter algorithm for stemming" had to be applied on the list of hyperwords (see in Table 2) in order to compute the stemmed words corresponding to each hyperword. The *vocabulary of hyperwords* in a document classification problem included all different hyperwords which appear in all the documents of the problem in question. The sizes of the *vocabularies of hyperwords* have been 642, 491, and 712 in problems 1, 2, and 3, respectively. We remark that the employment of hyperwords has resulted in a reduction of order 4:1 in the number of features.

For document classification the *s-FLN* neural network has been employed again, but this time the *s-FLN* was applied in the lattice "N-dimensional unit hypercube", where N is the vocabulary length in a document classification problem. For details of applying the *s-FLN* in the N-dimensional unit hypercube the reader may refer to [Kaburlasos97], [Petridis98], [Petridis99]. Since the classification performance of *s-FLN* depends on the order of data presentation [Petridis99] an ensemble of 9 *s-FLN* neural modules has been employed and, during testing, a datum was assigned to the category which received most "votes". Each *s-FLN* module has been trained using a different random permutation of the same training data set with vigilance parameter ρ=0.90. The time required to train one *s-FLN* module on a Pentium III at 500 MHz with 128 RAM was in the range 1 sec to 3 secs depending on the problem. The average number of hyperboxes calculated by an ensemble of 9 *s-FLN* modules is shown in Table 4. Hence, "words" resulted in an average of 13, 12.55, and 15.11 hyperboxes, respectively, in problems 1, 2, and 3, whereas "hyperwords" resulted in an average of 21.77, 17.22, and 27.55 hyperboxes in problems 1, 2, and 3, respectively. The larger number of hyperboxes, resulting in when "hyperwords" were employed has been attributed to the smaller dimension of the corresponding vectors of features; in particular, since the same value of the vigilance parameter ρ=0.90 was employed in both experiments, that is with words and with hyperwords, it follows that the maximum allowable hyperbox size is smaller with hyperwords than with words and proliferation of hyperboxes was implied in the experiments with hyperwords. Note that the relation between 1) the vigilance parameter ρ, and 2) a hyperbox size, during learning by *s-FLN* will be detailed in a future publication.

The document classification results are shown in Table 4 for the three problems. In particular, in problem-1 the employment of hyperwords resulted in better results than the employment of words, that is 81.03% versus 77.58%. In problem-2 it has been the employment of words, which resulted in better classification results, the corresponding percentages have been 95.74% versus 91.48%. Finally, in problem-3 the employment of hyperwords has provided with better classification results, in particular 96.55% versus 94.82%. It might be worthwhile noting that problems 1 and 3 involve longer documents than problem 2 as shown by the corresponding "Average" Statistics in Table 4. By experimental evidence

it seems that the differences in performance are related to the length of the corresponding documents in a problem. More specifically, the employment of words seems to favor document classification problems involving short documents, whereas the employment of hyperwords seems to favor document classification problems involving longer documents. An explanation might be that in shorter documents fewer semantically related words are expected than in longer documents, hence by replacing different words by the same hyperword in a short document is more prone to a steep deterioration in performance due to error.

TABLE 4

Statistics of three document classification problems and comparative performance in each problem using either words of hyperwords. The *s-FLN* neural network has been employed in the unit N-dimensional hypercube with vigilance parameter ρ=0.90. The average number of hyperboxes computed by an ensemble of 9 *s-FLN*'s is also shown. For longer documents the employment of hyperwords has demonstrated better classification results.

Problem	Number of Categories	Statistics of document length (no. of words)		Classification Performance Using			
				Words		**Hyperwords**	
		Average	Standard Deviation	Average no. of Hyperboxes	Success	Average no. of Hyperboxes	Success
1	3	71.79	51.59	13	77.58 %	21.77	81.03 %
2	2	37.58	34.47	12.55	95.74 %	17.22	91.48 %
3	2	95.77	51.11	15.11	94.82 %	27.55	96.55 %

It is useful to discuss comparatively document classification results obtained by other methods. Note, in the first place, that the computation of hyperwords in this work implies a *clustering* of (semantically related) words. Another method for clustering words based on a similarity of their corresponding probability distributions over the categories of a text classification problem has been reported in [Baker]. The method in question, namely *distributional clustering*, is shown to reduce, in a specific problem, the dimensionality of feature vectors by three orders of magnitude while performance remains 2% lower than the performance attained when the full vocabulary is used. Moreover, the document classification performance by "distributional clustering" is compared with performance by other feature selection algorithms based on information-gain, mutual-information, etc. The best document classification performances reported in [Baker] are in the area of 80% for different methods when a few thousand of features are used; nevertheless only distributional clustering is reported to retain 80% classification performance using even less than 100 features. Nevertheless a disadvantage of distributional clustering appears to be the long preprocessing time required for calculating the probability distribution of each individual word in a vocabulary of 62,258 words stemming from "20 Newsgroups" benchmark collection of documents for classification involving 20 categories and 20,000 documents; moreover the probability

distributions in question have to be re-calculated in practice should new documents for training become available. On the other hand, the calculation of hyperwords for document classification as described in this work needs to be carried out only once for all different document classification problems. Furthermore, the employment of hyperwords, as explained in this work, has demonstrated as well the potential to exceed the performance of using the full vocabulary as it has been reported in Table 4 for problems 1 and 3.

6 Discussion and Conclusion

A survey of six World Wide Web (WWW) search engines has shown that any one engine indexes less than about one-third of the "indexable Web", and the latter contained an estimated lower bound of 320 million homepages in early 1998 [Lawrence]. A more recent estimate in June 2000 brings the number of web pages up to more than one billion [Guersey]. There is a need to keep up with the accelerating pace of electronic document proliferation in various tasks including automated information retrieval, electronic routing, etc [Weiss].

A connectionist scheme, namely *s- Fuzzy Lattice Neurocomputing* (*s-FLN*) scheme has been presented in this work as a promising tool for document classification based on semantics. In particular, the *s-FLN* is characterized by a capacity to cluster directed graphs stemming from a master-graph. A master-graph has emanated from a Thesaurus of the English language synonyms, where a word in the Thesaurus corresponds to a node- and a synonym implies a link- in the master-graph in question. Experimental work reported here has demonstrated that the clusters computed by *s-FLN* are sets of semantically related words, namely hyperwords. The arithmetic parameters of *s-FLN* can be adjusted so as to tune the granularity of learning in a specific application. The utility of hyperwords has been demonstrated in three document classification problems, where the hyperwords have implied a 4:1 reduction in the number of features and, moreover, the hyperwords have demonstrated the capacity for good document classification performance. In this work a Thesaurus of the English language has been used, nevertheless the techniques presented here are applicable in principle with a Thesaurus of other natural languages.

The computation of hyperwords as well as their employment for efficient-, content-based document classification by a connectionist scheme amounts to an innovation. It should also be pointed out that in this work the contents of a specific node have been neglected during clustering because such contents do not exist. Hence clustering by *s-FLN* has been effected based solely on master-graph's topology rather than based, as well, on node contents. Nevertheless, the *s-FLN* can potentially accommodate the contents

of a node by employing an additional constituent lattice. Note that neurocomputing by *s-FLN* in a master-graph using both master-graph's topology and nodes' contents, might be useful in applications involving the World Wide Web (WWW) where the nodes might be web-pages and the links are hyperlinks between web-pages. The later is a topic for future research.

Appendix

Proof of Theorem 6

The following proof is valid for any complete lattice L with a positive valuation function v such that $v(O_{\mathsf{L}})=0$. It is shown underneath that function $s(x,u)=\frac{v(x\wedge_{\mathsf{L}} u)}{v(x)}$ satisfies the three conditions (*E0*) thru (*E2*) of definition 4, therefore $s(x,u)$ is an inclusion measure.

(*E0*) $s(x,O_{\mathsf{L}})=\frac{v(x\wedge_{\mathsf{L}} O_{\mathsf{L}})}{v(x)}=\frac{v(O_{\mathsf{L}})}{v(x)}=\frac{0}{v(x)}=0$, for $x\neq O_{\mathsf{L}}$. Note that $s(x,O_{\mathsf{L}})$ is guaranteed to be equal to 0 (for $x\neq O_{\mathsf{L}}$) only if v is a positive, and not merely a monotone, valuation in lattice L.

(*E1*) $s(u,u)=\frac{v(u\wedge_{\mathsf{L}} u)}{v(u)}=\frac{v(u)}{v(u)}=1$, $\forall u\in \mathsf{L}$. For $u= O_{\mathsf{L}}$, $s(u,u)$ is assumed to be equal to 1 by definition.

(*E2*) In any lattice L the operation of meet ($\wedge_{\mathsf{L}}$) is *monotone* [Birkhoff], that is, $u\leq_{\mathsf{L}} w \Rightarrow x\wedge_{\mathsf{L}} u \leq_{\mathsf{L}} x\wedge_{\mathsf{L}} w$, $x,u,w\in \mathsf{L}$. Hence, $u\leq_{\mathsf{L}} w \Rightarrow x\wedge_{\mathsf{L}} u \leq_{\mathsf{L}} x\wedge_{\mathsf{L}} w \Rightarrow v(x\wedge_{\mathsf{L}} u) \leq v(x\wedge_{\mathsf{L}} w) \Rightarrow \frac{v(x\wedge_{\mathsf{L}} u)}{v(x)} \leq \frac{v(x\wedge_{\mathsf{L}} w)}{v(x)} \Rightarrow$ $s(x,u) \leq s(x,w)$, for $x,u,w\in \mathsf{L}$ and $x\neq O_{\mathsf{L}}$.

∇

Acknowledgement

The authors gratefully acknowledge Thanasis Kehagias, first, for bringing to our attention the MOBY Thesaurus, and second, for his assistance in producing the *simplified MOBY Thesaurus*. We would also like to thank Pavlina Fragou for her assistance with the preprocessing of Reuters-21578 documents.

REFERENCES

[Baker] L.D. Baker, and A.K. McCallum, "Distributional Clustering for Text Classification", *Proc. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, 1998.

[Birkhoff] G. Birkhoff, *Lattice Theory*, Providence, RI: American Mathematical Society, Colloquium Publications, vol. 25, 1967.

[Carpenter87] G. Carpenter, and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics and Image Understanding*, vol. 37, pp. 54-115, 1987.

[Carpenter91] G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system", *Neural Networks*, vol. 4, pp. 759-771, 1991.

[Chang] C.-H. Chang, and C.-C. Hsu, "Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW", *IEEE Trans. on Knowledge and Data Engineering*, vol. 11, no. 4, pp. 595-609, 1999.

[Cohen] W.W. Cohen, "Learning Trees and Rules with Set-valued Features", *1996 AAAI Proc. Thirteen National Conference on Artificial Intelligence*, Portland, Oregon, 5 August 1996.

[Drucker] H. Drucker, D. Wu, and V.N. Vapnik, "Support Vector Machines for Spam Categorization", *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1048-1054, 1999.

[Frasconi] P. Frasconi, M. Gori, and A. Sperduti, "A General Framework for Adaptive Processing of Data Structures", *IEEE Trans. on Neural Networks*, vol. 9, no. 5, pp. 768-786, 1998.

[Georgiopoulos] M. Georgiopoulos, H. Fernlund, G. Bebis, and G.L. Heileman, "Order of Search in Fuzzy ART and Fuzzy ARTMAP: Effect of the Choice Parameter", *Neural Networks*, vol. 9, no. 9, pp. 1541-1559, 1996.

[Gori] M. Gori, F. Scarselli, "Are Multilayer Perceptrons Adequate for Pattern Recognition and Verification ?", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1121-1132, 1998.

[Green] S.J. Green, "Building Hypertext Links by Computing Semantic Similarity", *IEEE Trans. on Knowledge and Data Engineering*, vol. 11, no. 5, pp. 713-730, 1999.

[Guersey] L. Guersey, "The Search Engine as Cyborg", *New York Times* (06/29/00), P.E1.

[Huang] J. Huang, M. Georgiopoulos, and G.L. Heileman, "Fuzzy ART Properties", *Neural Networks*, vol. 8, no. 2, pp. 203-213, 1995.

[Junker] M. Junker, and A. Abecker, "Exploiting Thesaurus Knowledge in Rule Induction for Text Classification", in *Proc. Recent Advances in Natural Language Processing (RANLP'97)*, Tzigov Chark, Bulgaria, 11-13 September 1997, pp. 202-207.

[Kaburlasos97] V.G. Kaburlasos, and V. Petridis, "Fuzzy Lattice Neurocomputing (FLN) : A Novel Connectionist Scheme for Versatile Learning and Decision Making by Clustering", *International Journal of Computers and Their Applications*, vol. 4, no. 2, pp. 31-43, 1997.

[Kaburlasos00] V.G. Kaburlasos, and V. Petridis, "Learning and Decision-Making in the Framework of Fuzzy Lattices", in *New Learning Techniques in Computational Intelligence Paradigms*, L.C. Jain editor. Boca Raton, FL: CRC Press, 2000 (to be published).

[Lawrence] S. Lawrence, and C.L. Giles, "Searching the World Wide Web", *Science*, vol. 280, 3 April 1998, pp. 98-100.

[Lecun] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[Lewis] D.D. Lewis, Professional Home Page for David D. Lewis of AT&T Labs – Research <http://www.research.att.com/~lewis>.

[Mladenic98] D. Mladenic, "Machine Learning on Non-homogeneous, Distributed, Text Data", Ph.D. dissertation, Dept. Computer and Information Science, University of Ljubljana, Slovenia, 1998.

[Mladenic99] D. Mladenic, "Text-Learning and Related Intelligent Agents: A Survey", *IEEE Intelligent Systems*, vol. 14, no. 4, pp. 44-54, 1999.

[MOBY] MOBY Thesaurus, http://www.dcs.shef.ac.uk/research/ilash/Moby/mthes.html.

[Omlin] C.W. Omlin, and C.L. Giles, "Constructing Deterministic Finite-State Automata in Recurrent Neural Networks", *Journal of the ACM*, vol. 43. no. 6, pp. 937-972, 1996.

[Petridis98] V.P. Petridis, and V.G. Kaburlasos, "Fuzzy Lattice Neural Network (FLNN): A Hybrid Model for Learning", *IEEE Trans. on Neural Networks*, vol. 9, no. 5, pp. 877-890, 1998.

[Petridis99] V.P. Petridis, and V.G. Kaburlasos, "Learning in the Framework of Fuzzy Lattices", *IEEE Trans. on Fuzzy Systems*, vol. 7, no. 4, pp. 422-440, 1999.

[Plate] T.A. Plate, "Holographic Reduced Representations", *IEEE Trans. on Neural Networks*, vol. 6, no. 3, pp. 623-641, 1995.

[Pollack] J.B. Pollack, "Recursive Distributed Representations", *Artificial Intelligence*, vol. 46, nos. 1-2, pp. 77-106, 1990.

[Porter] M. Porter, The "official" home page for distribution of the Porter Stemming Algorithm <http://www.muscat.com/~martin/stem.html>.

[Rumelhart] D.E. Rumelhart, and J.L. McClelland, "On Learning the Past Tenses of English Verbs", in *Parallel Distributed Processing; Volume 2: Psychological and Biological Models*, J.L. McClelland, D.E. Rumelhart and the PDP Research Group. Cambridge, MA: MIT Press, 1986, pp. 216-271.

[Quinlan] J.R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1994.

[Sahami] M. Sahami, "Using Machine Learning to Improve Information Access", Ph.D. dissertation, Dept. Computer Science, Stanford University, 1998.

[Scarselli] F. Scarselli, and A.C. Tsoi, "Universal Approximation Using Feedforward Neural Networks: A Survey of Some Existing Methods, and Some New Results", *Neural Networks*, vol. 11, no. 1, pp. 15-37, 1998.

[Sperduti95] A. Sperduti, A. Starita, and C. Goller, "Learning Distributed Representations for the Classification of Terms", *Proc. Intl. Joint Conf. on Artificial Intelligence*, 1995, pp. 509-515.

[Sperduti97] A. Sperduti, and A. Starita, "Supervised Neural Networks for the Classification of Structures", *IEEE Trans. on Neural Networks*, vol. 8, no. 3, pp. 714-735, 1997.

[Touretzky] D. Touretzky, "BoltzCONs: Reconciling Connectionism with the Recursive Nature of Stacks", *Connectionist Symbol Processing*, G. Hinton, ed., Cambridge, MA: MIT Press, 1991.

[Weiss] S.M. Weiss, C. Apte, F.J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz, and T. Hampp, "Maximizing Text-Mining Performance", *IEEE Intelligent Systems*, vol. 14, no. 4, pp. 63-69, 1999.