Hierarchical Word Clustering - automatic thesaurus generation

Victoria J. Hodge [1]

Dept. of Computer Science, University of York, UK, vicky@cs.york.ac.uk

Jim Austin

Dept. of Computer Science, University of York, UK, austin@cs.york.ac.uk

Abstract

In this paper, we propose a hierarchical, lexical clustering neural network algorithm that automatically generates a thesaurus (synonym abstraction) using purely stochastic information derived from unstructured text corpora and requiring no prior word classifications. The lexical hierarchy overcomes the *Vocabulary Problem* by accommodating paraphrasing through using synonym clusters and overcomes *Information Overload* by focusing search within cohesive clusters. We describe existing word categorisation methodologies, identifying their respective strengths and weaknesses and evaluate our proposed approach against an existing neural approach using a benchmark statistical approach and a human generated thesaurus for comparison. We also evaluate our word context vector generation methodology against two similar approaches to investigate the effect of word vector dimensionality and the effect of the number of words in the context window on the quality of word clusters produced. We demonstrate the effectiveness of our approach and its superiority to existing techniques.

Key words: neural network, hierarchical thesaurus, lexical, synonym clustering.

1 Introduction

Due to the proliferation of information in databases and on the Internet, users are overwhelmed producing *Information Overload*. Web Search engines can return millions of potential matches to user queries even when more complex

[1] This research was supported by an EPSRC studentship.

(and user-unfriendly) Boolean logic is employed. Web search engines can be slow, although faster search engines are being developed, and matching is often poor (quantity does not necessarily indicate quality) as Search Engines often employ simple keyword pattern matching that takes no account of relevance. Search Engines often simply return the document with the greatest number of keyword occurrences. A methodology to process documents unsupervised, handle paraphrasing of documents, to focus retrieval by minimising the search space and to automatically calculate the document similarity from statistics available in the text corpus is desired. Document may be clustered according to the user's requirements (clustered 'on the fly') and then employ category-specific finer-grained matching techniques.

Word categorisation (encompassing both unsupervised clustering and supervised classification) enables the words to be associated or grouped according to their meaning to produce a thesaurus. In this paper we focus solely on word clustering as this approach is unsupervised. Clustering does not require pre-generated human classifications to train the algorithm and is therefore less subjective and more automated as it learns from text corpus knowledge only. Word clustering can also overcome the *Vocabulary Problem* cited by Chen et al. [2]. They posit that through the diversity of expertise and background of authors and the polysemy of language, there are many ways to describe the same concept; there are many synonyms. In fact, Stetina et al. [20] postulate that polysemous words occur most frequently in text corpora even though most words in a dictionary are monosemous. Humans are able to intuitively cluster documents from imputed similarity. They overcome the differing vocabularies of authors and the inherent synonymy and polysemy of language. A computerised system must be able to match this ability. For computerised document similarity calculation, an underlying hierarchical synonym clustering is required to enable differing vocabularies to be accommodated. The distances in the hierarchy may be used for word similarity estimation and to score document similarity, thus allowing paraphrased documents to be awarded high similarity scores as their contained words fall into identical or neighbouring synonym clusters. Human generated thesauri are too general; they encompass all senses of words even though many are redundant for a particular domain. They are expensive with respect to construction time particularly if a single human knowledge engineer generates the hierarchy. If multiple experts are consulted then it is very difficult to obtain a single unified hierarchy. Human thesauri also omit certain senses and subdivide others where there is little distinction; they are rather subjective. Automatic methods can be trained generally or domain specifically as required. The hierarchy allows us to focus searching to cohesive clusters therefore minimising the search space for each query. In this paper we analyse current word categorisation approaches and describe and evaluate our method with respect to the current implementations. We compare our TreeGCS clustering method [7], [6] and sections 3.2 and 3.3 to the Self-Organising Map (SOM) [11] method and then compare

three methods for context vector generation where the vector dimensionality and the number of words in the context are varied. We demonstrate the necessity of using high-dimensional vectors to represent the individual words in the documents. High dimensional vectors ensure that the word vectors are approximately orthogonal and there are no implicit word dependencies or relationships in the vectors representing the individual words. Therefore all dependencies and relationships are imputed purely from the relationships between the document words. We demonstrate the superiority of a wider context window when generating the context vectors, illustrating the superior quality clusters and higher stability of the cluster topology produced. Finally we establish the higher quality of the clusters produced by TreeGCS compared to SOMs. The clusters produced from TreeGCS are similar to the clusters extracted from a benchmark human generated thesaurus

Our approach is entirely automated and uses only unstructured text corpora as data. The motivation for our approach derives from the patterns present in text. These patterns produce statistical correlations in the context patterns of individual words. We can thus infer the similarities of words from their contexts, as similar words (synonyms) will have similar contexts due to their correlations. Through unsupervised text processing and clustering we represent semantic relationships by categorising the word co-occurrence patterns. We do not need to generate any linguistic structures, which are complex to produce and tend to be susceptible to linguistic ambiguities. We automatically infer a domain-specific or generalised hierarchical thesaurus as required. We therefore surmount the *Vocabulary Problem* [2] by permitting synonym retrieval to match paraphrased documents. We can use the thesaurus to award scores to synonyms using the intra-cluster distances and the inter-cluster distances in the hierarchy.

2 Current Methods

Current approaches for textual analysis are multifarious and diverse. The motivations encompass word sense disambiguation, synonym inferencing and both classification and clustering. They include (the following list is not exhaustive but is intended to be broad): contextual methods, WordNet hierarchy methods, clustering methods and SOM methods.

2.1 Contextual Methods

These approaches utilise the local neighbourhood of words in a document (the context) to establish lexical similarity and impute synonym groups or disam-

biguate polysemic words. **Yarowsky** [22] employs two phases: an iterative bootstrapping procedure and an unsupervised categorisation phase. All instances of a polysemous word are identified in the text corpus. A number of representative samples are selected from each sense set and used to train a supervised classification algorithm. The remainder of the sense sets are trained into the supervised classifier. The classifier may additionally be augmented with one sense per discourse information, i.e., document topic. The classifier can then be used in an unsupervised mode to categorise new exemplars. Stetina et al [20] postulate that one sense per discourse holds for nouns but evidence is much weaker for verbs. The approach therefore is only suitable for nouns and requires an appraisal of the text corpus before processing commences to identify the nouns. The method is only partially unsupervised requiring a supervised initial training method; i.e. human intervention which can be time consuming.

The motivation for **Shütze & Pederson** [19] is a lexical hierarchy exploiting contextual statistics and requiring no prior data knowledge. The algorithm collects a symmetric, term-by-term matrix recording the number of times that words i and j co-occur in a symmetric window centred about word i in the text corpus, where i and j are any random word indices from the list of all corpus words. Singular-valued decomposition (SVD) is used to reduce the dimensionality of the matrix to produce a dense vector for each item that characterises its co-occurrence neighbourhood. The dense co-occurrence vectors are clustered using an agglomerative clustering algorithm to generate a lexical hierarchy. The method groups words according to their similarity unsupervised rather than some pre-computed thesaurus categories. However, vector dimensionality reduction introduces computational complexity and may cause information loss as the vectors induced represent the meta-concepts and not individual words. Shannon's Theory states that *the more infrequent a word the more information it conveys.* These may well be discarded by SVD. The method does not account for the proximity of the word co-occurrences (co-occurrence is considered from a purely binary perspective). There is no weighting of the co-occurrence according to the two terms' proximity in the context window.

2.2 WordNet Hierarchy

These methods utilise the human-generated hierarchical categorisation of synonyms, hyponyms (IS-A) and metonyms (PART-OF) of WordNet to estimate word similarity and the most appropriate word sense (WordNet lists all senses of words with the most frequently occurring listed first). **Li, Szapakowicz & Matwin's** [14] method utilises the WordNet synonym, hyponym and metonym hierarchy to assign word similarity according to the distance in the hierarchy. Similarity is inversely proportional to distance. However, the distance of tax-

onomic links is variable, due to certain sub-taxonomies being much denser than others. Again the technique relies on an underlying predetermined word hierarchy and can only process words present in the hierarchy; it could not extrapolate similarities to new words. Human generated thesauri are subjective and rely on sense categorisation decisions made by the human constructor.

2.3 Clustering

An unsupervised clustering algorithm derives the word clusters and models of association directly from distributional data rather than pre-determined classes as in Yarowsky. **Pereira, Tishby & Lee** [17] employ a divisive clustering algorithm for probability distributions to group words according to their participation in particular grammatical relations with other words. In the paper, nouns are classified according to their distribution as direct objects of verbs with cluster membership defined by $p(c|w)$ (the probability a word belongs to a cluster) for each word rather than hard Boolean classification. Deterministic annealing finds the sets of clusters by starting with a single holistic cluster and increasing the annealing parameter (see paper [17]). As the annealing parameter increases, the clusters split producing a hierarchical data clustering. The approach is limited to specific grammatical relations, requiring a pre-processor to parse the corpus and tag the part-of-speech. At the time of writing, the authors felt the technique required further evaluation.

2.4 Self-Organising Map (SOM) Methods

Word vectors or document vectors form the input vector space of the SOM [11] to permit topological mapping, to infer similarity and categorise words and documents. The aim of **Lowe** [15] is a topological mapping of contextual similarity exploiting contextual information to derive semantic relationships. Each word in a 29-word vocabulary is associated with a 58-element co-occurrence vector. The value of the nth attribute in the co-occurrence vector reflects the number of times the nth word of the vocabulary has preceded and the $(n + 29)$th attribute represents the number of times the nth word has succeeded the keyword where $1 \leq n \leq 29$. The 58 element vectors form the input vectors for a SOM network. The SOM is labelled by determining the best matching unit for each input vector. The word contexts (labels) are arranged topologically according to lexical and semantic similarity by the SOM. However, the method is inherently susceptible to the scalability problem; vector length grows linearly in relation to lexical size and thus the method is not feasible for a large vocabulary.

Ritter & Kohonen's [18] approach provides the motivation for our system. A topological map of semantic relationship among words is developed on a self-organising feature map. In the initial implementation, each word has a unique, seven-dimensional, unit length vector assigned. The input vector space is formed from the average context in which each word occurs in the text corpus. Semantic similarity is induced from context statistics, i.e., word neighbourhoods using a window of size three, one word either side of the target word, (only nouns verbs and adverbs are used in the method). The method has been extended to WEBSOM [9], [8], [12] that categorises over one million documents using a window size of three and 90-dimensional word vectors. The approach is entirely unsupervised requiring no human intervention and parallelisable enabling computational speedup. However, SOMs cannot form discrete (disconnected) clusters thus inhibiting the data representation. The clusters have to be determined after the algorithm terminates by hand and this introduces the innate subjectivity of human judgements. Also, the word topography in WEBSOM is single-layered compared to the hierarchical topology we induce using TreeGCS [7] [6] and described later in the paper.

2.5 Summary

Many of the methods exposited use Zipf's Law [23] and stop-word elimination to reduce the vocabulary of the text corpus to be processed, some even implement word stemming. Zipf's Law implies that a significant portion of the words in a corpus constitutes the words that appear most infrequently whereas frequently occurring words comprise a relatively small portion of the corpus. Many approaches eliminate these infrequent words to decrease vector dimensionality and computational requirements. The designers of these approaches deem that such words provide little discriminatory power for document similarity assessment. We feel that this may discard essential information. Although we do not generate context averages for frequent words, e.g., {the, and, but, etc.}, we include these words in the context averaging of the keywords. For this reason we use a context size of seven (three words either side of the target word). We demonstrate in sections 4 and 6 the qualitative improvement of word clustering against a human-generated thesaurus and Euclidean distance-based vector approach of a size seven window compared to size three. Ritter & Kohonen [18] and their extrapolations [9], [8], [12], fix the context window at three and thus have to discard frequent terms, infrequent terms and punctuation etc. We feel these provide much information and are certainly employed by a human reader when parsing text. Dagan, Lee & Pereira [3] empirically demonstrated that singleton words (words occurring once) were important for parsing concurred by Shannon's theory. An infrequent word occurring only once in two documents may be the key to identifying those documents and should not be discarded from the indexing.

The larger window allows us to maximise the lexical information used and minimise the amount of pre-processing required.

3 Our Methodology

We cluster words into a synonym hierarchy using the TreeGCS hierarchical clustering neural network that we have developed, described in [7] [6] and sections 3.2 and 3.3. TreeGCS is an unsupervised growing, self-organising hierarchy of nodes able to form discrete clusters. Similar high-dimensional inputs are mapped onto a two-dimensional hierarchy reflecting the topological ordering of the input vector space. We assume a latent similarity in word co-occurrences and use TreeGCS to estimate word similarity from contextual statistics without resorting to a human-generated thesaurus. We categorise all keywords as discussed previously and perform no dimensionality reduction thus decreasing information loss. The process is fully automated, requires no human intervention or data processing as the context vectors are generated automatically from unstructured text data and the clustering requires minimal knowledge of the data distribution due to the self-organising network. Each node in the hierarchy represents a small group of synonyms at the lowest level and progressively larger groups of related words up through the tree. The distance between the nodes in the tree is directly proportional to the similarity of the word sets they represent.

3.1 Pre-processing

All upper-case letters are converted to lower-case to ensure matching. A list of all words and punctuation marks in the text corpus is generated and a unique random, real-valued, unit-length m-dimensional vector $\vec{x}$ is assigned to each word as in (equation 1).

$$Word \rightarrow \vec{x} \in \Re^m \tag{1}$$

Stop-words are removed to create a second list of keywords. A moving window of size n is passed across the text corpus, one word at a time (see figure 1). Ritter & Kohonen use a context window of size three, we use size seven and illustrate the qualitative improvement this generates in sections 4 and 6. If the word in the window centre is a keyword ($\vec{x}^{middle} \in \{keyword\}$) then the unique, random, real-valued, unit length m-dimensional vectors representing each word in the window of size n ($\vec{x^1} \ldots \vec{x^n}$) are concatenated and added to the $m * n$ dimensional context vector $\vec{y}_{keyword}$ representing the keyword (equation 2).

$$\vec{y}^{keyword} \in \Re^{m*n} = \vec{y}^{keyword} + \vec{x^1} \ldots \vec{x^n} \wedge (\vec{x}^{middle} \in \{keyword\}) \tag{2}$$

When the entire corpus has been processed, all context vectors generated for each keyword are averaged (total for each dimension / frequency of keyword), see equation 3.

$$\vec{Avg}^{y} = \forall i \frac{\vec{y}_i}{frequency} \tag{3}$$

$$\vec{Avg}^{y} = symFact * \vec{y}_i \text{for keyword attributes} \tag{4}$$

The keyword attributes in the average vector are finally multiplied by the symbol factor ($symFact$ in equation 4). The keyword is multiplied by a symbol factor of value 0.2 in Ritter & Kohonen's method for average context vector generation and also in the WEBSOM average context vector generation technique. The symbol factor diminishes the relative influence of the keyword (the central word in the context window) in relation to the surrounding words in the context window for the average context vectors. This prevents the actual keyword over-influencing the topological mapping formation and places the emphasis for topology and semantic similarity inferral on the context vector attributes, the surrounding words. We empirically determined the optimum factor value for our approach with a context window size of seven and found for seven-dimensional vectors that a symbol factor of 0.4 produced the optimal cluster quality (as judged by the authors). For 90-dimensional vectors the symbol factor has far less influence over the Euclidean distances between the context averages and thus the clusters generated, so we elected to use a symbol factor of 0.4, as this was more effective for the seven-dimensional vectors. This prevents the keyword over-influencing the context average but still provides sufficient influence for a context window of size seven.

As with Deerwester [4], we handle synonymy but only partially accommodate polysemy. Polysemic words are again represented by a weighted average of their contexts but we only generate one context for polysemic words (the context is the mean context of all word senses biased by the frequency of occurrence of each sense). For example, *plant* may be *a living organism* or *heavy machinery*. Only one context average would be produced for *plant*.

3.2 GCS Algorithm

Our TreeGCS method is based on the Growing Cell Structure (GCS) method that is described next and is adapted from [5]. GCS networks form discrete clusters unlike SOMs where the SOM cells remain connected in a lattice structure. The dimensions of the SOM lattice have to be pre-specified (such as the 9x9 grid used in our evaluation later in this paper). Contrastingly only the maximum number of cells needs to be pre-specified in GCS and the network grows dynamically by adding new cells and deleting superfluous cells until the maximum number of cells is reached. The initial topology of GCS is a 2-dimensional structure (triangle) of cells (neurons) linked by vertices. We

use a 2-dimensional cell network to allow a hierarchy to be superimposed and to allow visualisation if necessary. Each cell has a *neighbourhood* defined as those cells directly linked by a vertex to the cell. The input vector distribution is mapped onto the cell structure by mapping each input vector to the best matching cell. Each cell has a $contextWindow * wordVectorDimensionality$-dimensional vector attached denoting the cell's position in the input vector space; topologically close cells have similar attached vectors. On each iteration, the attached vectors are adapted towards the input vector. The adaptation strength is constant over time and only the *best matching unit (bmu)* and its direct topological neighbours are adapted unlike SOMs where the adaptation occurs in a progressively reducing radius of neurons around the *bmu*. Cells are inserted where the cell structure under-represents the input vector distribution and superfluous cells that are furthest from their neighbours are deleted. Each cell has a '*winner counter*' variable denoting the number of times that cell has been the *bmu*. The winner counter of each cell is reduced by a predetermined factor on every iteration. The aim of the GCS method is to evenly distribute the winner counter values so that the probability of any cell being a *bmu* for a random input is equal, i.e., the cells accurately represent the input space.

The GCS learning algorithm is described below, the network is initialised in point 1 and points 2 to 7 represent *one iteration*. An *epoch* constitutes one iteration (points 2 to 7) for each input vector in the dataset, i.e. one pass through the entire dataset.

(1) A random triangular structure of connected cells with attached vectors ($w_{c_i} \in \Re^n$) and E representing winner counter (the number of times the cell has been the winner) is initiated.
(2) The next random input vector ξ is selected from the input vector density distribution. The input vector space is represented as real-valued vectors of identical length.
(3) The best matching unit (bmu) is determined for ξ and the *bmu*'s winning counter is incremented.
$bmu = ||\xi - w_c||_{min_{c \in network}}$ where $||\ ||$ = Euclidean distance
$\Delta E_{bmu} = 1$
(4) The best matching unit and its *neighbours* are adapted towards ξ by adaptation increments set by the user.
$\Delta w_{bmu} = \epsilon_{bmu}(\xi - w_{bmu})$
$\Delta w_n = \epsilon_i(\xi - w_n)\ \ (\forall n \in neighbourhood)$
(5) If the number of input signals exceeds a threshold set by the user a new cell (w_{new}) is inserted between the cell with the highest winning counter (w_{bmu}) and its farthest *neighbour* (w_f) - see figure 2,
The weight of the new unit is set according to:
$w_{new} = (w_{bmu} + w_f)/2$.
Connections are inserted to maintain the triangular network configuration. The winner counter of all *neighbours* of w_{new} is redistributed to

donate fractions of the neighbouring cells' winning counters to the new cell and spread the winning counter more evenly,
$\Delta E_n = -\frac{1}{|n|}E_n \ \ (\forall n \in neighbourhood\ of\ w_{new})$.
The winner counter for the new cell is set to the total decremented from the winning counters of the neighbouring cells.
$E_{new} = \sum(\frac{1}{|n|}E_n \ \ (\forall n \in neighbourhood\ of\ w_{new})$.

(6) After a user-specified number of iterations, the cell with the greatest mean Euclidean distance between itself and its *neighbours* is deleted and any cells within the neighbourhood that would be left 'dangling' are also deleted (see figure 3). Any trailing edges are deleted to maintain the triangular configuration.
$Del = max_{c \in network}(\frac{\sum ||w_c - w_n||}{card(n)} \forall n \in neighbourhood)$

(7) The winning counter variable of all cells is decreased by a user-specified factor to implement temporal decay.
$\Delta E_c = -\beta E_c \ \ \forall c \in network$

The user-specified parameters are: the dimensionality of GCS which is fixed at 2 here, the maximum number of neighbour connections per cell, the maximum cells in the structure, ϵ_{bmu} the adaptation step for the winning cell, ϵ_i the adaptation step of the neighbourhood, β the temporal decay factor, the number of iterations for insertion and the number of iterations for deletion.

The algorithm iterates until a specified performance criterion is met, such as the network size. If the maximum number of epochs and the maximum number of cells are specified as the termination criteria then new cells are inserted until the maximum number of cells is reached. Once the maximum has been reached, adaptation continues each iteration and cells may be deleted. The cell deletion reduces the number of cells to below the maximum allowing one or more new cells to be inserted until the maximum number of cells is reached again. Deletion removes superfluous cells while creating space for new additions in under-represented regions of the cell structure so the input distribution mapping is improved while the maximum number of cells is maintained.

3.3 TreeGCS Algorithm

The TreeGCS is superimposed onto the standard GCS algorithm exposited above. A tree root node points to the initial cell structure and incorporates a list of all cells from the GCS. As the GCS splits or clusters are deleted, the tree divides and removes leaf nodes to parsimoniously summarise the disjoint network beneath and the GCS cell lists are updated with each leaf node holding a list of all GCS cells in its associated cluster. Only leaf nodes maintain a cluster list. A parent's cluster list is implicitly a union of the children's cluster lists

and is not stored for efficiency - minimising memory usage. No constraints are imposed on the tree hence it is dynamic and requires no prior data knowledge - the tree progressively adapts to the underlying cell structure. The hierarchy generation is run once after each GCS epoch. The running time per hierarchy generation iteration is $O(cells)$ as we essentially breadth-first search through the entire cell structure.

A conceptual hierarchy of word synonym clusters is generated. The distance in the hierarchy between two concepts is inversely proportional to the similarity. Concepts are progressively more general and the cluster sets become larger towards the root of the hierarchy.

The underlying GCS's algorithm is susceptible to the ordering of the input vector space, if we alter the order of the input vectors in the dataset, a different cluster topology is generated for each unique input vector order [7]. Thus, in TreeGCS we only commence cell deletion once 90 % of the total cells required in the cell structure have been added [7]. This delayed deletion prevents premature cluster committal and ensures the GCS network has evolved sufficiently before cluster splitting commences. In addition, we also iterate between different orders of the input vector space to ameliorate the order susceptibility (the x dimensional vectors that represent the context averages are rearranged to generate different data orders). Iterating between different orders cancels out the variance in the hierarchical structures generated by the different orders, vastly improving the algorithm's qualitative performance. The algorithm for the tree superimposition is detailed below in pseudocode.

For each epoch,
  Execute the GCS epoch, forming an unconnected graph representing the disjoint clusters.
  Breadth first search from the final winning cell for the epoch to determine which cells are present in the cluster.
  While some cells remain unprocessed,
    Breadth first search from the next unprocessed cell to determine which cells are present in the cluster.
  If the number of clusters has increased from the previous epoch, **then** any tree nodes that point to multiple clusters are identified and child nodes are added for each new cluster formed (see figure 4). The cluster list of the parent is deleted and cluster lists are updated for the child nodes. If a cluster is formed from new cells (cells inserted during the current epoch) then a new tree node is added as a child of the root and the new cluster cells added to the new node's list.
  Elsif the number of clusters has decreased, a cluster has been deleted and the associated tree node is deleted. The tree is tidied to remove any redundancy (see figure 5).
  For each unprocessed cluster, the tree node that points to that cluster is

determined, the cluster list emptied and the new cells are added.

The GCS cells are labelled, see figure 6. Each input vector is input to the GCS and the cell identifier of the *bmu* is returned. The cell can then be labelled with the appropriate word. Words that occur in similar contexts map to topologically similar GCS cells thus reflecting syntactic and semantic similarity through purely stochastic background knowledge. Tree nodes are merely pointers to GCS cells. All nodes except leaf nodes have only an identifier and pointers to their children. The leaf nodes have an identifier but also point to the GCS cell clusters and implicitly the GCS cell labels (they maintain a list of the identifiers of the GCS cells in their respective clusters). When the GCS *bmu* is identified, the associated tree node can also be identified and the tree can be traversed to find all word distances from the distances between the clusters (leaf nodes) in the tree.

4 Evaluation

We initially demonstrate the qualitative effectiveness of our average vector generation method against the R & K and WEBSOM approaches. We also demonstrate the qualitative effectiveness of our TreeGCS algorithm against the SOM algorithm. Human clustering is innately subjective. In an experiment by Macskassy et al. [16], no two human subjects produced 'similar' clusters when clustering the information contained in a set of Web pages. This creates difficulties for cluster set evaluation and determining whether computerised methods are qualitatively effective. We evaluate the quality of the three methodologies for average context vector generation by comparing the TreeGCS clusters produced from the vectors for each methodology against a dendrogram cluster set produced from the same vectors to provide a Euclidean distance-based evaluation. We then compare the TreeGCS hierarchies against the cluster sets of a human-generated thesaurus. We compare TreeGCS and SOM clustering by evaluating the topologies produced for each set of average context vectors against the clusters produced by a dendrogram trained on the same vectors. Fritzke has previously demonstrated GCS's superior performance with respect to correctly classified test patterns over 6 common neural network approaches and the nearest neighbour statistical classifier for mapping the vowel recognition dataset [5]. In this paper, we use a small dataset comprising 51 words to enable visualisation of the cluster structures and cluster contents and to permit a qualitative comparison of the cluster structures and cluster contents. A larger dataset would preclude visualisation of the cluster structures as they would be too complex to draw and a qualitative comparison of the cluster structures generated would thus be extremely difficult for a larger dataset.

4.1 Three Methods for Context Vector Generation

We emulate the **Ritter & Kohonen methodology** as faithfully as possible. We remove common words, punctuation and numbers from the text corpus. We select the vectors from a distribution of random numbered, seven dimensional vectors. We use a context window of size three. We multiply the keyword vector by a symbol factor of 0.2. The following cluster topologies were generated from the text corpus using words that occurred ten times or more. We chose to only cluster word frequency 10 words to ensure the context vectors were truly averaged and not biased by limited exposure and also to eliminate infrequent terms as R & K.

WEBSOM, the new development of the R & K approach, uses 90-dimensional real-valued random vectors for the words. Kaski [10] showed that the orthogonality is proportional to vector dimensionality and we have found that for seven-dimensional vectors, the actual vector assigned to each word in the corpus affects the context averages and thus the similarity and clustering produced. The seven-dimensional approach is also more susceptible to the symbol factor as the multiplier has more effect on the Euclidean distance than for 90-dimensional where the effect is less. WEBSOM extends R & K and uses 90-dimensional word vectors, context window of size three and symbol factor 0.2.

Our methodology described in section 3 varies slightly from the previous two. We only remove numbers from the corpus, the previous two methods also remove common words and punctuation. We do not generate context averages for common words and punctuation but use them in the context window of other words, hence we have a larger context window of size seven. Our method uses 90-dimensional vectors and symbol factor 0.4. Again only words occurring ten or more times are shown in the clusters to ensure the contexts were averaged and not biased by infrequency due to the small size of our test corpus. Although normally we would include these words, we wanted a valid comparison to the previous methods.

We use the Ritter & Kohonen method, WEBSOM method and our method for average context vector generation to produce three sets of vectors. We train each of the three average context vector sets in turn into a standardised benchmark Euclidean distance-based clustering algorithm, the **dendrogram**, to derive three TreeGCS and three dendrogram clusterings, one for each context vector generation method. We compare the TreeGCS and dendrogram cluster topologies for each vector methodology. We produced one 25x25 matrix for each average vector methodology indexed by the 25 most similar words from the dendrogram. The 25 words are shown in bold text in figures 7, 9 and 11. For each TreeGCS cluster (figures 7, 9 and 11) we placed a 1 at position ij

in the respective matrix if $word_i$ and $word_j$ co-occured in a TreeGCS cluster otherwise we entered a 0 in the matrix if they were in different clusters. We filled the half matrix where $i < j$ so there was only one entry per ij pair to prevent redundancy as co-occurrence is symmetric, if ij co-occur then ji must co-occur. After completing each matrix, we counted the number of 1s entered. This represents the number of words clustered together in the dendrogram top 25 cluster that are also clustered together in the TreeGCS cluster structure generated from each vector methodology. The highest score indicates that the vectors generated from that method enable the TreeGCS to most closely emulate the Euclidean distance-based cluster of the dendrogram. The three TreeGCS clusters in figures 7, 9 and 11 vary in depth so for a consistent comparison we reduced the depth of the WEBSOM and our TreeGCS trees to level 4 shown in figures 9 and 11 which is equivalent to the shallowest hierarchy by merging all clusters below level 4 to form a single leaf cluster at level 4.

We further compare each of the three TreeGCS clusters to a human word hierarchy derived from the MS Bookshelf[2] thesaurus. We produced a 51x51 matrix for the thesaurus clusters and the three TreeGCS hierarchies, indexed by all 51 words in the evaluation. Again we entered a 1 at position ij in the respective matrix if $word_i$ and $word_j$ co-occured in a cluster otherwise we entered a 0 in the matrix. Again, we filled the half matrix where $i < j$. This produces 4 matrices. We overlaid each of the three TreeGCS matrices in turn over the human matrix and counted the number of 1s in the same position in both matrices. This represents the number of words clustered together in both the human and TreeGCS cluster structures, the higher the score, the more similar the TreeGCS clusters are to the human clusters. Again we repeated the evaluation with the WEBSOM and our TreeGCS hierarchies limited to level 4 for consistency with the R & K hierarchy.

4.2 TreeGCS versus SOM Clustering Comparison

We then train the three sets of average context vectors generated by the three methods into a SOM and TreeGCS for comparison of the accuracy of the two clustering algorithms. We compare TreeGCS versus SOM purely on vector distances by analysing the distribution of the dendrogram words (the 25 closest words with respect to Euclidean distance) through the TreeGCS and SOM clusters. The dendrogram cluster sets act as benchmarks, to ensure that the mapping of input vectors to cells for both TreeGCS and SOMs are preserving

[2] We were unable to use the WORDNET hierarchy as it does not contain all of the words from the text corpus. This precludes the use of synSet distances from the WORDNET hierarchy [14] (described previously) as an evaluation tool. Bookshelf allows us to generate clusters distances but no word similarity distances.

the vector distances. We validate that the TreeGCS clusters more accurately emulate the dendrogram cluster sets.

4.3 Text Corpus, Dendrogram and Thesaurus

The text corpus for the evaluation was taken from the economic data in the World Factbook [21] for each of the countries in Europe. This corpus is written in correct English, the vocabulary is reasonably small allowing a compact thesaurus to be generated with many words that have similar meanings allowing the cluster quality to be readily evaluated. We cluster the context averages of the words that occur ten times or more in the text corpus for all evaluations. This emulates the R & K and WEBSOM methodologies that remove infrequent terms and it maintains a consistency of words to be clustered to ensure a valid and consistent comparison.

The dendrogram hierarchically illustrates similarities and is ideal for structure comparison. The dendrogram uses the centroid-clustering algorithm where the algorithm iteratively merges clusters. Initially there is one data point per cluster. Each cluster is represented by the average of all its data points, the mean vector; the inter-cluster distance is defined as the distance between pairs of mean vectors. The algorithm iteratively determines the smallest distance between any two clusters, (using the Euclidean distance metric) and the two clusters are merged producing a branch in the cluster tree. The merging is repeated until only one cluster is left. However, dendrograms have problems with identical similarities as only two clusters may be merged at each iteration, so if there are two pairs of clusters with equal distances, one pair has to be merged on one iteration and the other pair on the next iteration, the order being arbitrary. In dendrograms, visualisation is difficult for a large dataset as there is one leaf node for each data point so it is very difficult to view more than 500 data points. Therefore we feel a dendrogram would be unsuitable as the underlying mechanism for a lexical clustering method but is relevant for structure and cluster comparisons on a small dataset. Both TreeGCS and SOMs use Euclidean distance when mapping the inputs on to the output topology so we feel the dendrogram is consistent with these approaches. Each input vector is represented by a leaf node in the dendrogram. In the SOM and TreeGCS, many vectors can map to leaf nodes so we can use the comparison with the dendrogram to ensure the vector mappings are not distorted and Euclidean distance-based vector similarities preserved when multiple vectors map to leaf nodes.

We produced synonym sets from the MS Bookshelf thesaurus to allow comparisons of the clusters generated in our evaluations with a human generated clustering. The synonym sets are arranged in similarity order, the closer to-

gether the more similar and the greater the distance the more dissimilar the words. The synonym sets are:

- {economy, system, market, budget, policies, program, government, account}
- {investment, resources, welfare, privatization, reform}
- {output, energy, exports, gdp, trade}
- {industry, agriculture}
- {growth, progress, inflation}
- {debt, deficit},
- {economic, financial, industrial, monetary}
- {agricultural}
- {currency}
- {capita, percent, sector}
- {substantial, large, highly}
- {small}
- {foreign, private, public}
- {countries, republic, state}
- {european, eu, europe, union, western}
- {unemployment }
- {years}

4.4 Settings

All settings are summarised in tables 1, 2 and 3. Table 1 compares the settings for the generation of the averaged context vectors from the word contexts in the text corpus for each of the three methods evaluated. Table 2 compares the settings for the SOM for each method of word context vector generation. We use the SOM-PAK [13] SOM implementation (as used in WEBSOM [8]). We use the parameter settings that produced the minimal quantisation error for a 9x9 map of rectangular topology, using the neighbourhood kernel 'bubble', (where the neighbourhood function refers to the set of array points around the node). WEBSOM required a different setting for α (the cell vector adaptation parameter) compared to the other two methods to minimise the quantisation error of the topological mapping from the input space to the 9x9 map. Table 3 compares the settings for the TreeGCS for each method of word context vector generation. We set the parameters to produce the 'best' quality clusters as judged by the authors, see [7] for a discussion of selecting parameter combinations. The seven-dimensional vector evaluation required different parameters from the 90-dimensional trial.

5 Results

We detail the cluster topologies produced by TreeGCS, SOM and the dendrogram for each context vector methodology.

5.1 Ritter & Kohonen Method

- From the dendrogram clustering using the R & K average context vectors, the 25 most similar words are:
  {system union output industry substantial policies exports european privatization countries sector inflation percent economic foreign financial resources government growth large economy unemployment gdp years eu}. These words index the 25x25 matrix used to evaluate the average context vector methodologies.
- For the TreeGCS hierarchy generated using the R & K average context vectors see figure 7. The words in bold are the 25 most similar words identified by the dendrogram generated using the R & K average context vectors and are used to form the 25x25 matrix with ij set to 1 where $word_i$ and $word_j$ are in the same TreeGCS cluster.
- For the SOM cluster topology (see figure 8), again the 25 most similar words from the dendrogram are highlighted in bold.

5.2 WEBSOM

- From the dendrogram generated using the WEBSOM average context vectors, the 25 most similar words are:
  {countries european budget exports industry sector agricultural industrial large system trade output gdp financial eu economic economy government growth inflation percent privatization unemployment years foreign}. These words form the indices for the 25x25 matrix.
- See figure 9 for the TreeGCS hierarchy generated using the WEBSOM average context vectors. The words in bold are the 25 most similar words identified by the dendrogram generated using the WEBSOM average context vectors.
- For the SOM, the topology is illustrated in (see figure 10), again the 25 most similar words from the dendrogram are highlighted in bold and form the co-occurrence entries in the 25x25 matrix.

5.3 Our methodology

- For the dendrogram generated using average context vectors produced by our method, the cluster of the 25 most similar terms is:
  { small reform percent exports gdp output system agricultural market budget industrial financial foreign large industry privatization inflation growth economic eu economy government trade unemployment energy}. These words index the matrix.
- For the TreeGCS hierarchy generated from average context vectors produced by our method see figure 11. The words in bold are the 25 most similar words identified by the dendrogram generated from the average context vectors produced by our method. The co-occurrence statistics form the matrix entries.
- For the SOM the cluster topology is shown in figure 12. The 25 most similar words from the dendrogram are shown in bold. We have also included the Sammon mapping (see [13]) for the SOM (See figure 13). The Sammon mapping maps the n-dimensional input vectors onto 2-dimensional points on a plane.

6 Analysis

6.1 Three Methods for Context Vector Generation

From table 4, the TreeGCS cluster produced from our method for average context vector generation is most similar to both the dendrogram and the human cluster sets. When we reduce the TreeGCS structure to level 4 for equality with the shallowest TreeGCS structure, our vector generation method is even more similar to both the dendrogram and human clusters.

The TreeGCS structures generated from 90-dimensional vectors emulate human clusterings more closely than dendrograms from the seven-dimensional vectors. The higher dimensionality vectors increase word vector orthogonality; a prerequisite for the 'bag of words' average context vector generation approach. It is imperative that the vectors ascribed to the individual words in the text corpus imply no ordering of the words so text processing is based purely on the processing of sequences of words. For seven dimensional vectors the Euclidean distances are altered between the context averages when different vectors are initially assigned to different words. This is particularly important for low frequency words where the context average is biased by the vectors assigned. Even for words occurring greater than 10 times, the vector

assignment influences the similarities. Kaski [10] showed that there is a direct correlation between vector dimensionality and orthogonality - the higher the dimensionality the greater the orthogonality. We empirically evaluated various dimensionalities for consistency with respect to cluster content when different vectors are initially ascribed to the words in the corpus. We used the dendrogram to pinpoint the most similar 25 words. We found that the cluster sets were identical for 90-dimensional vectors over a set of experiments but varied for all dimensionalities tested below 90. The higher dimensionality spreads the vectors more across the input pace allowing a more accurate differentiation of clusters. We feel similar methods, using self-organising maps or growing cell structures, should use vectors of this dimensionality or greater to ensure orthogonality and spread and to maintain consistency and stability of the lexical clusters regardless of initial word-vector assignments.

With respect to the size of the context window, we feel that our size seven-context window produces superior quality TreeGCS clusters to WEBSOM's context window of size three. The TreeGCS clusters produced from the average context vectors produced by our method emulate both the nearest neighbour (dendrogram) and human generated thesaurus more accurately than the TreeGCS cluster produced from the WEBSOM average context vectors. The vast majority of terms from the dendrogram and MS Bookshelf are in the three clusters (see figure 11) for our vector generation method but are spread across four clusters with many of the other words also within these clusters for the WEBSOM method of vector generation (see figure 9).

6.2 TreeGCS versus SOM Clustering Comparison

For all three evaluations in sections 5.1, 5.2 and 5.3, the top 25 words from the dendrogram are spread across the SOM (as can be seen from the spread of bold text in figures 7, 9 and 11) but tend to be in closely related clusters in the TreeGCS hierarchy with just the odd exception (the bold text occurs in clusters that are near neighbours in the hierarchy). For example, for our method (see figure 11), the dendrogram words, shown in bold text, are predominantly in three clusters and these clusters are very closely related. Only 'industry' and 'unemployment' are clustered elsewhere. With respect to Euclidean distance, the TreeGCS emulates the nearest neighbour approach of the dendrogram far better than the SOM. The Sammon mapping produced from the SOM using our method to derive the context vectors is extremely distorted (see figure 13). SOMs are criticised in the literature [1] for distorting high dimensional inputs when they map onto the 2-dimensional representation.

7 Conclusion and Future Work

We feel that our method, 90-dimensional vectors, symbol factor of 0.4, context window of seven is superior to the R & K and WEBSOM methods. Our method for context vector generation enables TreeGCS to be more similar to both the dendrogram (Euclidean distance) and the human generated thesaurus than either the R & K or WEBSOM approaches. We note that the corpus effects the similarity of the computer generated structures against a human thesaurus. The human thesaurus encompasses general word meanings while the text corpus may be very specific so the similarity of the computer generated approaches to the human clusters is affected and may appear low. We also demonstrated that the TreeGCS algorithm emulates Euclidean vector-distance based cluster sets more faithfully than the SOM algorithm. Therefore, we feel the optimum approach for synonym clustering of the methods evaluated is to generate the average context vectors using our method and train these in to the TreeGCS cluster algorithm. TreeGCS not only emulates the nearest neighbour and human generated clusters more faithfully, it forms discrete clusters and dynamically forms a lexical hierarchy.

There are two main drawbacks to our current method. The first is the inability to disambiguate words. All senses of a polysemic word are averaged together during the context average formation, distorting the averaged context vectors produced. We intend to improve this by including part-of-speech tagging to differentiate identical words which represent different parts-of-speech, for example *spring: noun, a water source* and *spring: verb, to jump*. However, autonomously differentiating word senses is currently intractable and relies on a knowledge engineer to tag the senses.

The second main drawback lies in the underlying GCS algorithm and is a speed problem. The algorithm is dependent on the winner search - finding the best matching unit. This involves comparing the input vector to the vector attached to each cell, calculating the difference for each vector dimension. This must be repeated for each vector in the input vector space to complete each epoch. This search is therefore, (number of input vectors * vector dimensionality * number of cells) $\approx O(n^3)$ for each GCS epoch. For the small vocabulary evaluated in this paper the speed problem was not apparent. However, for a large corpus, with an extensive vocabulary the speed is slow and we need to speed the algorithm, reduce the running time and hence remove the bottleneck.

8 Acknowledgement

We would like to thank the anonymous reviewers of our original draft for their incisive remarks and recommendations which allowed us to improve the overall presentation of the paper.

References

[1] C. M. Bishop. *Neural networks for pattern recognition.* Oxford, Clarendon P., 1995.

[2] H. Chen, C. Schuffels, and R. Orwig. Internet Categorization and Search: A Machine Learning Approach. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):88–102, 1996.

[3] I. Dagan, L. Lillian, and F. C. N. Pereira. Similarity-Based Methods for Word Sense Disambiguation. In *35th Annual Meeting of the Association for Computational Linguistics*, San Francisco, California, 1997. Morgan Kaufmann.

[4] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*, 1(6):391–407, 1990.

[5] B. Fritzke. Growing Cell Structures - a Self-organizing Network for Unsupervised and Supervised Learning. Technical Report TR-93-026, International Computer Science Institute, Berkeley, CA, 1993.

[6] V. Hodge and J. Austin. Hierarchical Growing Cell Structures: TreeGCS. In *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems, August 30th to September 1st*, 2000.

[7] V. Hodge and J. Austin. Hierarchical Growing Cell Structures: TreeGCS. *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Connectionist Models for Learning in Structured Domains*, 2001.

[8] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM–self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[9] T. Honkela, V. Pulkki, and T. Kohonen. Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. In F. Fogelman-Soulie and P. Gallinari, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN-95)*, volume 2, pages 3–7. EC2 et Cie, Paris, 1995.

[10] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of IJCNN'98, International Joint Conference on Neural Network*, volume 1, pages 413–418, IEEE Service Center, Piscataway, NJ, 1998.

[11] T. Kohonen. *Self-Organizing Maps*, volume 2. Springer-Verlag, Heidelberg, 1997.

[12] T. Kohonen. Self-organization of very large document collections: State of the art. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, volume 1, pages 65–74. Springer, London, 1998.

[13] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. Som_pak, the self-organizing map program package.

[14] X. Li, S. Szpakowicz, and S. Matwin. A WordNet-based Algorithm for Word Sense Disambiguation. In *Proceedings of IJCAI-95*, Montréal, Canada, 1995.

[15] W. Lowe. Semantic Representation and Priming in a Self-organizing Lexicon. In *Proceedings of the 4th Neural Computation Psychology Workshop*, pages 227–239. Springer Verlag, 1997.

[16] S. A. Macskassy, A. Banerjee, B. D. Davison, and H. Hirsh. Human Performance on Clustering Web Pages: A Preliminary Study. In *The Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

[17] F. Pereira, N. Tishby, and L. Lee. Distributional Clustering of English Words. In *Proceedings of ACL-93*, Columbus, Ohio, 1993.

[18] H. Ritter and T. Kohonen. Self-Organizing Semantic Maps. *Biological Cybernetics*, 61:241–254, 1989.

[19] H. Schütze and J. O. Pederson. Information Retrieval Based on Word Senses. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, NV, 1995.

[20] J. Stetina, S. Kurohashi, and M. Nagao. General Word Sense Disambiguation Method Based on a Full Sentential Context. In *Coling-ACL '98 Workshop "Usage of WordNet in Natural Language Processing Systems"*, Université de Montréal, Montréal, Canada, Aug. 1998.

[21] . World FactBook. http://www.odci.gov/cia/publications/factbook/country-frame.html.

[22] D. Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics. Cambridge, MA,*, pages 189–196, 1995.

[23] G. K. Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology.* Addison-Wesley, Cambridge, MA, 1949.

9 Tables

9.1 Table of the methodology settings

Method	Vector Dimensionality	Context Window	Symbol Factor
R+K	7	3	0.2
WEBSOM	90	3	0.2
Our's	90	7	0.4

Table 1
Table comparing the settings for the context vector generation in each of the three methods evaluated.

9.2 Table of the SOM settings

Method	α for x epochs	Radius for x epochs	α for next y epochs	Radius for next y epochs
R+K	0.9 for 3000	10 for 3000	0.5 for 27000	3 for 27000
WEBSOM	0.75 for 3000	10 for 3000	0.5 for 27000	3 for 27000
Our's	0.9 for 3000	10 for 3000	0.5 for 27000	3 for 27000

Table 2
Table comparing the parameter settings for the SOM algorithm to generate the map for each of the three context vector generation methods evaluated. α is the initial learning rate parameter which reduces to 0 during training and the *radius* is the neighbourhood of cells that are adapted in the SOM adaptation phase. The radius iteratively reduces to 0 during training.

9.3 Table of the TreeGCS settings

Method	ϵ_{bmu}	ϵ_i	β	Cells	Max Conns	Insertion	Deletion	Epochs
R+K	0.1	0.01	0.001	81	25	10	810	30000
WEBSOM	0.02	0.002	0.0002	81	25	10	810	30000
Our's	0.02	0.002	0.0002	81	25	10	810	30000

Table 3
Table comparing the parameter settings for the TreeGCS algorithm to generate the cluster hierarchy for each of the three context vector generation methods evaluated. N.B. Conns is an abbreviation for connections.

9.4 Table of the evaluations

Method	Dendrogram	Dendrogram Level 4	Human	Human Level 4
R+K	70	70	10	10
WEBSOM	88	100	22	26
Our's	93	253	32	74

Table 4
Table comparing the TreeGCS clusters produced from each of the three vector generation methodologies against the dendrogram and human generated clusters. We produced NxN matrices of all words to be clustered: the 25 most similar words from the dendrograms for the dendrogram comparison and the 51 cluster words for the human comparison. If two words ($word_i$ $word_j$) co-occur in a cluster then we inserted a 1 in the respective matrix otherwise we inserted a 0. We then counted the number of 1s in the 25x25 matrix for each vector methodology, where a word pair co-occur in both the dendrogram cluster and the TreeGCS cluster for that vector methodology. The counts are listed in column 2. We reduced all trees to level 4 shown in figures 7, 9 and 11 for equality and repeated the evaluation with the counts listed in column 3. We overlaid the human 51x51 matrix against each of the three 51x51 TreeGCS matrices and counted the number of overlaid 1s where a word pair co-occur in both the human and TreeGCS clusters, listed in column 4. Again we repeated the evaluation for the level 4 trees and the counts are given in column 5.

10 Captions

Figure 1: Figure illustrating the moving word window. The initial capital letter will be converted to lower case to ensure the 'he's match. Both instances of 'he' are represented by the same vector. The vectors associated with each word are concatenated to form the context vector for the target word 'he'.

Figure 2: Figure illustrating cell insertion. A new cell and associated connections are inserted at each step.

Figure 3: Figure illustrating cell deletion. Cell A is deleted. Cells B and C are within the neighbourhood of A and would be left dangling by the removal of the five connections surrounding A so B and C are also deleted.

Figure 4: Figure illustrating cluster subdivision. One cluster splits to form two clusters and the hierarchy is adjusted. The leftmost cluster then splits again.

Figure 5: Figure illustrating cluster deletion. The rightmost cell cluster is

deleted during an epoch (step 2) - this leaves a dangling pointer. The node with the dangling pointer is removed (step 3), leaving redundancy in the hierarchy. The redundancy is removed in the final step.

Figure 6: The cells in the GCS layer are labelled with the words they represent.

Figure 7: Ritter & Kohonen Methodology - TreeGCS cluster. The figures in bold indicate the top 25 words selected by the dendogram in each cluster.

Figure 8: Ritter & Kohonen Methodology - SOM mapping. The words in bold indicate the top 25 words selected by the dendogram

Figure 9: WEBSOM Methodology - TreeGCS cluster. The figures in bold indicate the top 25 words selected by the dendogram in each cluster.

Figure 10: WEBSOM Methodology - SOM mapping. The words in bold indicate the top 25 words selected by the dendogram

Figure 11: Our Methodology - TreeGCS cluster. The figures in bold indicate the top 25 words selected by the dendogram in each cluster.

Figure 12: Our Methodology - SOM mapping. The words in bold indicate the top 25 words selected by the dendogram

Figure 13: Figure illustrating the Sammon map generated for 90 dimensional vectors with context window = 7.

11 Figures

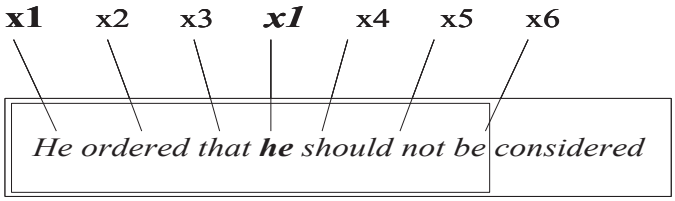

Fig. 1.

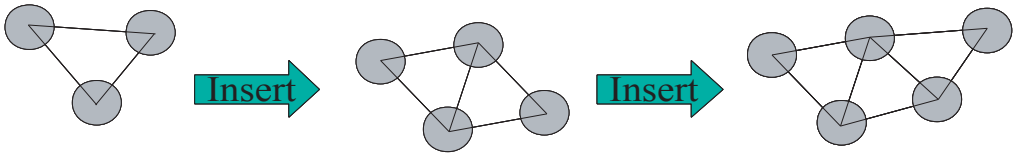

Fig. 2.

Fig. 3.

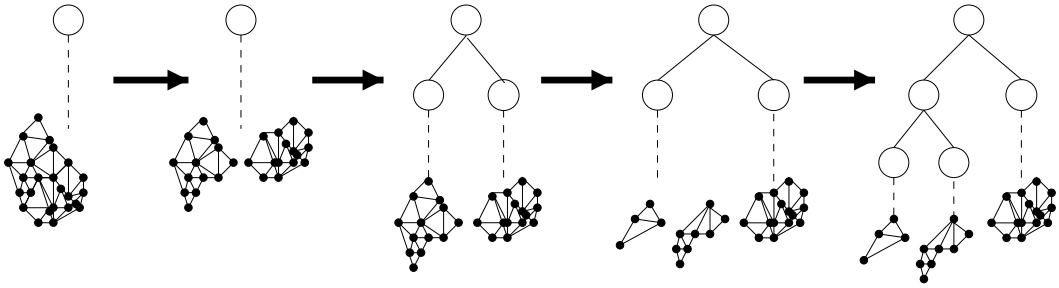

Fig. 4.

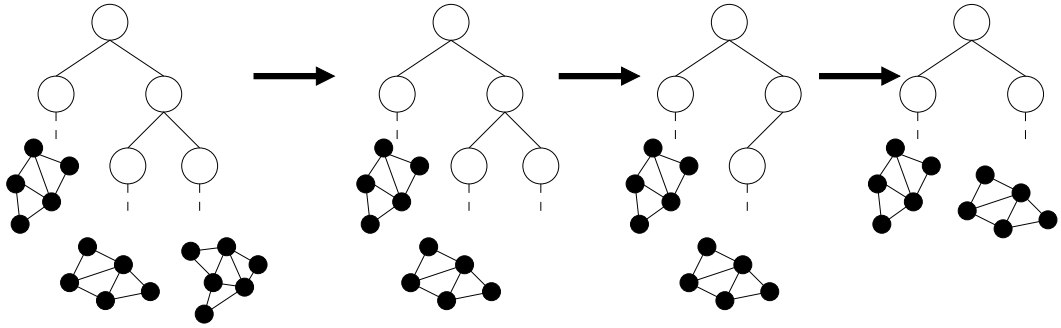

Fig. 5.

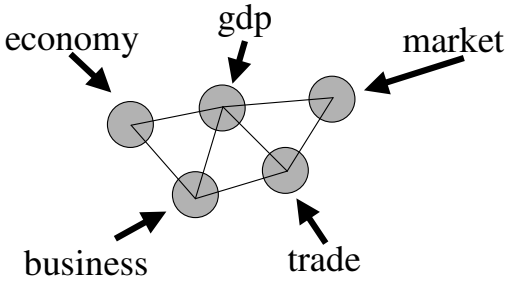

Fig. 6.

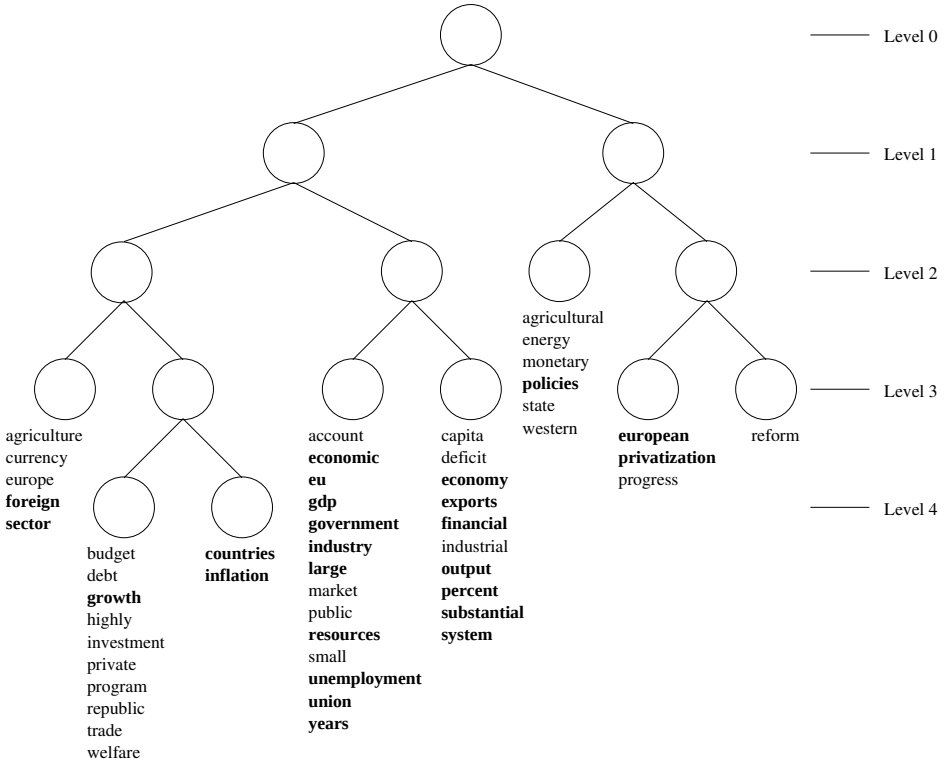

Fig. 7.

monetary		program		**inflation** **output**		republic trade		
	reform		**economy** **growth** **system**		**percent** **substantial**		budget welfare	
		deficit				highly **sector**		agriculture currency market
	industrial		**eu** **member**		europe investment			
european **privatization** progress		**financial**		account		debt		
economic			**government** public		**exports** **industry** **union**			
		large **resources**		**gdp** **years**		agricultural **policies**		private
			foreign small		low **unemployment**		state	
capita						**countries** energy		western

Fig. 8.

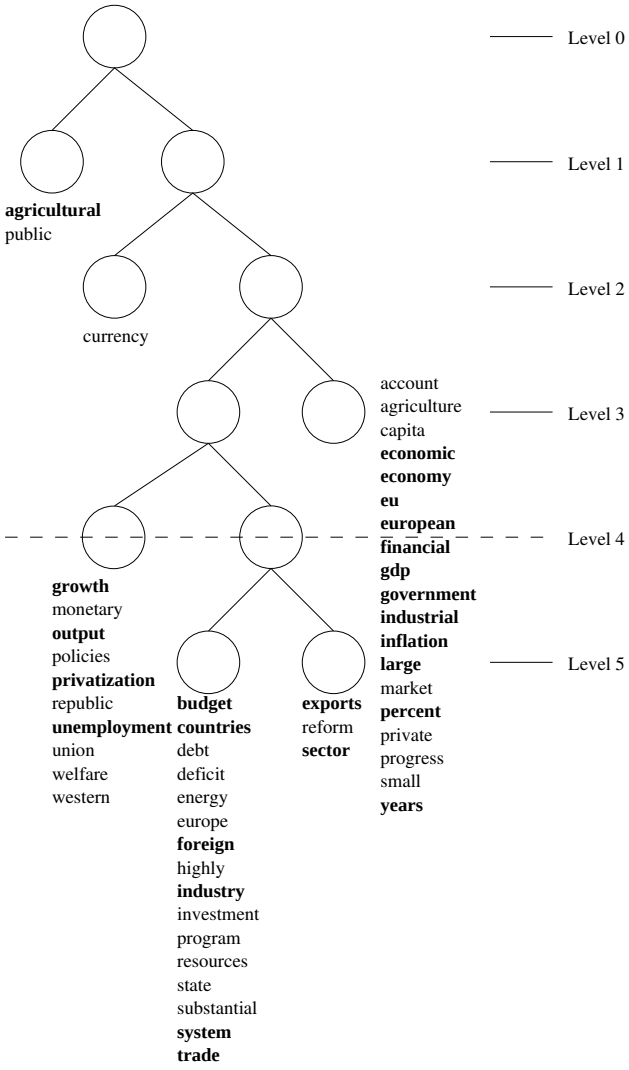
Level 0
Level 1
Level 2
Level 3
Level 4
Level 5
agricultural
public
currency
account
agriculture
capita
economic
economy
eu
european
financial
gdp
government
industrial
inflation
large
market
percent
private
progress
small
years
growth
monetary
output
policies
privatization
republic
unemployment
union
welfare
western
budget
countries
debt
deficit
energy
europe
foreign
highly
industry
investment
program
resources
state
substantial
system
trade
exports
reform
sector

Fig. 9.

capita		deficit				progress		agriculture market welfare
	eu						**industrial**	
currency debt investment **trade**		**exports** republic **sector**		**european** **large**		**growth** **percent** **privatization**		**agricultural** public
	economy		**economic**		**government**		**financial**	
foreign resources				**gdp** **output** **unemployment**		energy **industry**		program
	budget				**inflation**			
account union		**countries** europe		highly **system**		private small substantial		years
			state					
		policies		western		low		monetary reform

Fig. 10.

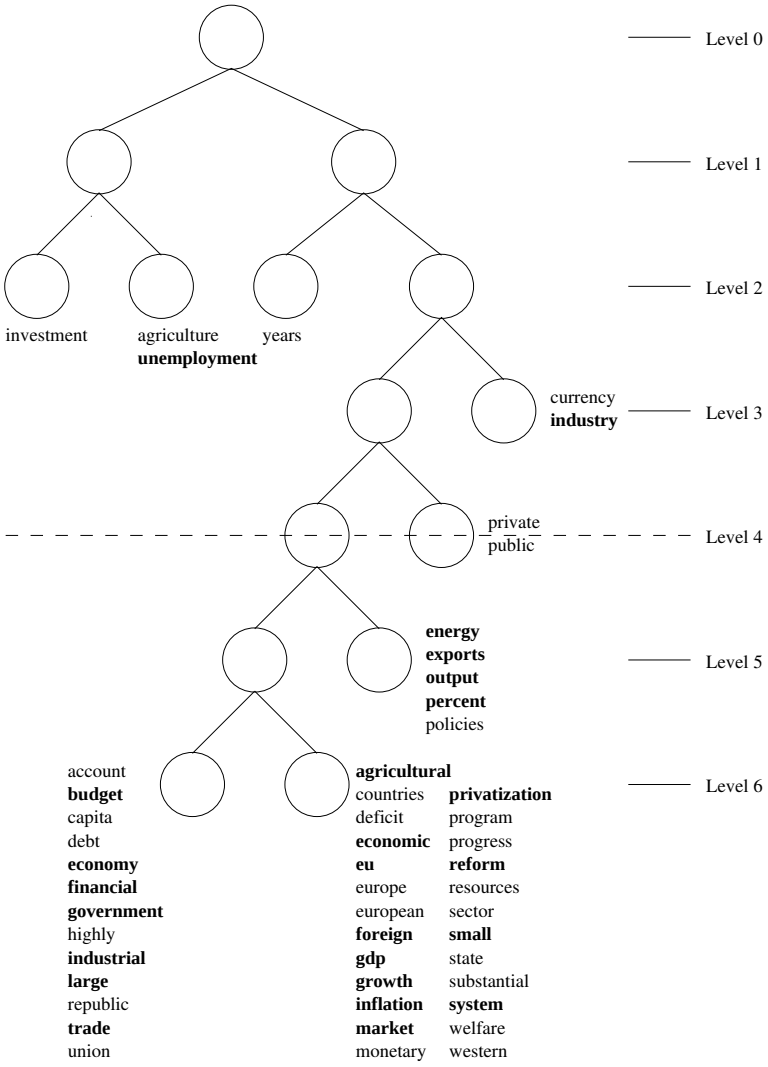
Level 0
Level 1
Level 2
investment
agriculture
unemployment
years
currency
industry
Level 3
private
public
Level 4
energy
exports
output
percent
policies
Level 5
account
budget
capita
debt
economy
financial
government
highly
industrial
large
republic
trade
union
agricultural
countries
deficit
economic
eu
europe
european
foreign
gdp
growth
inflation
market
monetary
privatization
program
progress
reform
resources
sector
small
state
substantial
system
welfare
western
Level 6

Fig. 11.

debt deficit republic sector		public		**large** substantial		**eu**	european	
					currency			**government**
percent years		countries europe		resources **small**		market		**budget** **economy**
			growth		**energy**		**industrial**	
policies progress		**privatization** **reform**		**exports**		**gdp** **output**		**agricultural** **financial**
			inflation		**trade**		state	
industry program		**system**		agriculture **unemployment**		**economic** private		**foreign** low
					highly		welfare	
monetary		investment		account union		capita		western

Fig. 12.

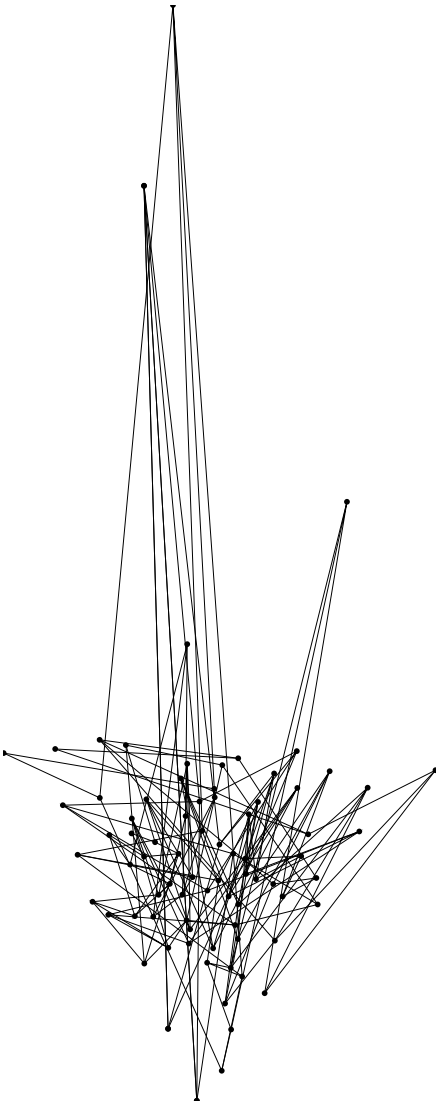

Fig. 13.

Contact Addresses:

Victoria Hodge
Department of Computer Science
University of York
York
UK
YO10 5DD
vicky@cs.york.ac.uk
Tel: +44 1904 432729
Fax: +44 1904 432767

Professor Jim Austin
Department of Computer Science
University of York
York
UK
YO10 5DD
austin@cs.york.ac.uk
Tel: +44 1904 432734
Fax: +44 1904 432767

Brief Biosketches of the authors:

Professor Jim Austin has the Chair of Neural Computation in the Department of Computer Science, University of York, where he is the leader of the Advanced Computer Architecture Group. He has extensive expertise in neural networks as well as computer architecture and vision. Jim Austin has published extensively in this field, including a book on RAM based neural networks.

Victoria Hodge is a PostGraduate Research Student in the Department of Computer Science, University of York. She is a member of the Advanced Computer Architecture Group investigating the integration of neural networks and information retrieval.