

Some Ways of Visualizing Results of Cluster Analysis

Deliverable 16 of the *Modus Operandi* project
carried out by the JRC for the
European Anti-Fraud Office OLAF

Administrative arrangement 14408-98-10

March 2000

Johan Hagman

**Joint Research Centre of the European Commission
Institute for Systems, Informatics and Safety (ISIS)
Risk Management and Decision Support Unit (G3)
Anti-Fraud Information Management (AIM)**

T.P. 361
21020 Ispra (VA), Italy
tel / fax + 39-0332-785646 / 9098
johan.hagman@jrc.it
<http://www.jrc.cec.eu.int/jrc>

Some Ways of Visualizing Results of Cluster Analysis

1	Introduction	1
2	Input format and characteristics of cluster analysis	1
3	Output format and characteristics of some visualization methods	1
3.1	Proximity lists	1
3.1.1	N closest neighbour lists of the data objects	1
3.1.2	N closest neighbour lists of the data parameters	2
3.1.3	Displaying ‘orthogonal characteristics’ for list items and hyperlinking	3
3.2	Dendrograms or ‘tree diagrams’	4
3.2.1	Displaying ‘orthogonal characteristics’ in dendrograms and hyperlinking	4
3.3	Item charts	6
3.4	Porting to other visualization software	7
3.4.1	The Link Notebook™ by i2 Limited	7
3.4.2	GIS systems	8
3.4.3	ThemeScape	8
4	Summary	8

1. Introduction

This report constitutes deliverable #16 of Modus Operandi, a project aiming at the automatic indexing and clustering of fraud-related texts. As a result of cluster analysis, documents are grouped together based on their similar vocabulary. The central topic of this specific report is not so much how to do cluster analysis in this context, but rather how to present, or visualize, the results of cluster analysis. To understand *what* is being visualized, the next section discusses the input data to be clustered. Documents, keywords and relevant vocabulary are all introduced there. The third and main section discusses some commonly used presentation methods like proximity lists, dendrograms, item charts &c, complete with examples from IRENE texts. Please, note that *some parts* of this report are aimed at the technical reader, e.g. an analyst who will be called to execute clustering and visualization tasks.

2. Input and characteristics of cluster analysis processing

Cluster analysis (CA) typically imports data in one of two formats where the second type may be an elaboration of the first. Let us explain this: the first data format is a table where N objects are described by M parameters – an ordinal orthogonal table, in mathematical terms. In the next step of CA, the parameters of this table are *normalized* and/or *equalized*¹ in order to make the values of all parameters comparable and usable for the clustering algorithm. Note that these parameters may very well refer to different scales and even scales of different types, i.e. using binary, integer, or float numbers. After this normalization/equalization, we will find each *object* described by a ‘value profile’ consisting of a vector of *parameter* values. These profiles can, in yet another step, be compared with each other whereby another table can be created containing the calculated *proximities* between each pair of objects/profiles. There are several algorithms for calculating this proximity value (also called ‘similarity’ or ‘distance’). We will henceforth refer to the result of this as the *proximity matrix*. So, from an object \times parameter table we will come up with an object \times object matrix. By a similar procedure we can also create a parameter \times parameter matrix. In that case we compare the parameters, (using *correlation analysis* or some other algorithm) i.e. how similarly they describe the objects in the original table.

Applied to the textual data in our example, the *objects are documents* and the *parameters are keywords*. The values in the table cells are occurrences of a given keyword in a given document. Instead of *absolute number* of occurrences, however, we use the more elaborate indicator *keyness value* which is a more accurate measure of the “importance” of that keyword to that text. The corresponding matrices will then be the document \times document matrix and the keyword \times keyword matrix and this is that second data format, mentioned above, which the cluster analyzer can accept as well, if such pre-calculated matrices are available. The document \times document matrix contains all information of how the documents form groups by virtue of common keyword(value)s but it is normally hard to detect these groups just by looking at the matrix. This is where CA is used to make evident *what* groups we have in our document collection. And, as the reader expects by now, virtually the same thing can be done with the keyword \times keyword matrix; the application of CA will show us what are the inherent grouping tendencies among them. In the next section we will see just how the result of applied CA may be presented.

3. Output and characteristics of some visualization methods

3.1 Proximity lists

3.1.1 N closest neighbour lists of the data objects

In one of our experiments we analyzed a set of ~1,500 documents, each of which is described by an average of ~15 keywords out of a set of ~1,800 keywords. The document names, or ‘identifiers’, indicate their different origin and *one* possible purpose of applying CA on this data set could be to see whether these documents cluster according to their origin (due to their having similar keywords typical of that origin). A quite trivial thing to extract from the document \times document matrix is, given any document, a ranked list of its most similar documents, together with some measure of similarity. This list could, of course, be exhaustive but normally some of the highest-ranking documents suffice. Figure 1 shows an extract from such a *proximity list* where each document identifier is listed with those of its five most similar “co-documents”. The similarity measure is relativized, i.e. it is expressed as percentage of the highest similarity value found to hold between any two documents in the collection once all *identical* documents have been detected and sorted out as duplicates.

¹ A. K. Jain & R. C. Dubes (1988) *Algorithms for Clustering Data*, Prentice Hall, Advanced Ref. Series.

FSBE96_002_FS_0		FSUK95_003_FD_0		FSUK95_004_FS_0	
FSBE96_003_FS_0	527	bureau	4	bureau	4
FSBE96_001_FS_0	367	cleveland	1	cleveland	1
FSBE96_004_FS_0	276	communication_trimestrielle	1	communication_trimestrielle	1
FSBE97_004_FO_0	128	conversation	1	conversation	1
FSUK96_001_FO_0	83	date_02_1996	1	date_03_1996	1
FSDE95_001_FO_0		décision_administrative	1	décision_administrative	1
FSDE94_002_FO_0	242	département_administratif	1	département_administratif	1
FSDE92_001_FS	98	détectif	1	détectif	1
FSSEL95_033_FS_0	84	détection	1	détection	1
FSSEL95_032_FS_0	84	état_membre	2	état_membre	2
FSDE95_007_FO_0	74	financement	2	financement	2
FSSES94_001_FO_0		fraude	1	fraude	1
FSSES96_009_FD_0	32	marais	1	marais	1
FSSES96_009_FS_0	29	police	3	police	3
FSSES96_007_FD_0	26	politique_structurelle	1	politique_structurelle	1
FSSES96_002_FD_0	24	présomption	1	présomption	1
FSSES96_003_FD_0	21	reglem_1681/94	3	reglem_1681/94	3
FSUK95_003_FD_0		reglem_1831/94	3	reglem_1831/94	3
FSUK95_004_FS_0	1000	sauvage	1	sauvage	1
FSUK97_003_FS_0	178	taxe	1	taxe	1
FSUK97_002_FS_0	111	uk00	4	uk00	4
FSUK95_002_FD_0	98	uk2	1	uk2	1
FSUK97_001_FS_0	94	uk60	1	uk60	1
		uk67	1	uk67	1

Figure 1 Top of a “5-closest-neighbour” list. **Figure 2** Profiles of the two documents found to be the most similar non-identical pair of the list in Fig. 1 where the maximum similarity equals 1,000.

There is yet no need for any CA to generate² the list of lists shown in Figure 1. Still we may say that it shows how the five most similar documents “cluster” around the one in focus and it is quite useful to be able to examine even this “monodimensional” clustering tendency around a given document. In the short extract in Figure 1 we see e.g. that the FSBE96_002_FS_0 document is most similar to the one presumably written right after (FSBE96_003_FS_0), second most similar to the document written just before (FSBE96_001_FS_0), and third most similar to an even later document (FSBE96_004_FS_0) in the same series. Examining the neighbours of the next document listed, we see that it is considerably more similar to another document belonging to the same ~FO~ series than it is to the subsequent documents. The third document listed is only weakly connected to other documents; the one it resembles the most has a similarity value which is just 3.2% of the highest value found between any two documents – and that pair is actually what we see next, under the fourth listed document. In a session of data exploration the explorer would probably be curious about these two documents and when examining these closely s/he will detect that they differ only in one keyword pair – which are nevertheless related to each other, i.e. the document FSUK95_003_FD_0 refers to the month of February 1996 and the document FSUK95_004_FD_0 refers to March the same year. The writer of the March report, it appears, copied the February report and just changed the date and if this is the case, it would also be a useful discovery as it might suggest revisions of the management routines in order to mitigate the information overload.

3.1.2 *N* closest neighbour lists of the data parameters

What we have done with the *objects*, i.e. the *documents*, so far in this experiment on clustering textual data, the cluster analyzer³ implemented at the ISIS/G3 section at JRC also lets us do with the *parameters*, i.e. the *keywords*. Figure 3 contains the corresponding proximity lists, or “5-closest-neighbour lists” for five of the approximately 1,800 keywords used. As a matter of fact, there are two lists for every keyword, in Figure 1 the documents are compared internally in terms of common keywords which they are *described by*, and in the column to the left in Figure 3, the keywords are compared internally in terms of common documents which they are *describing themselves*. The second column, or list, for each keyword in Figure 3 is a study of ‘synonymity’/‘complementarity’/‘interchangeability’ which, we suspected, could be calculated by looking for keyword pairs where both keywords appear *separately* together with a very similar set of other keywords but they never appear *together*, describing the same document. Similar experiments have been done by other researchers⁴ where “normal” words are clustered according to common concordance context and those results are very interesting as they fairly well reflect the ten-or-so parts of speech we use in grammars for that language – and this only by the words’ statistically similar distribution. Now, we are dealing not with “normal” words but key-

² Although there is a need of other essential processing steps, i.e. *duplicate detection*; *automatic keyword weight calculation and assignment* for all keywords in each document profile; *creation of the documentxdocument matrix* based on these weights, and *calculation of a ranking list* of neighbours for each document of which the top *N* (here five) neighbours are printed, creating the list shown in Fig. 1.

³ See J. Hagman (2000) *An Implemented Cluster Analyzer for Documents and their Indexing Terms*, Modus Operandi deliverable 12a, carried out by the JRC for the European Anti-Fraud Office OLAF.

⁴ At the time and place of writing this report this linguistic reference was unfortunately not available.

words, so we did not expect exactly the same result in our study but nevertheless we did detect a slight tendency in this direction. The words `annoncer` and `annonce` appear in a very similar context but virtually never together which might suggest they can be used interchangeably (which they also can, just altering the syntax a little as one of them is a verb and the other is the corresponding noun). In the same way `foins` appears with a set of other keywords (in its first column) which is very much the same set with which `tomate`, `fruit` and `prune` appear (as they are also being dried) but these three never or very seldom appear together with `foins`. In the case of `olive` we see that it appears very often with `huile`, and `olive` is also the keyword by far most strongly associated to `huile` (which we verify elsewhere in the list). Still, the other words in the left column of `olive` are also very relevant for each of the keywords `légumes ... tomate`, listed to the right, and this is likely to be so because these products are handled and sold in a similar fashion but very seldom mentioned with the word `olive` itself. We have already stated that by this comparably rough but cheap statistical method applied on the data we cannot expect pure synonyms in the second column in Figure 3 but the words we do find are often interesting anyway. In the case of the last keyword listed there, `unterhaltsgeld`, the keyword having the most similar context, `übernachtungskosten`, is, for example, a *hyponym* rather than a synonym.

<code>annoncer</code>			
<code>unterhaltsgeld</code>	571	<code>annonce</code>	626
<code>révocation</code>	548	<code>malversation</code>	591
<code>de00</code>	541	<code>surélevé</code>	587
<code>politique_structurelle</code>	520	<code>übernachtungskosten</code>	571
<code>installation</code>	519	<code>gesamtvollstreckungsverfahren</code>	562
<code>décision_administrative</code>			
<code>département_administratif</code>	965	<code>journal</code>	176
<code>communication_trimestrielle</code>	947	<code>omission</code>	160
<code>violation</code>	882	<code>demande_de_remboursement</code>	130
<code>politique_structurelle</code>	693	<code>de00</code>	130
<code>uk00</code>	686	<code>indemnité</code>	123
<code>foins</code>			
<code>fourrage</code>	987	<code>tomate</code>	219
<code>sécher</code>	950	<code>peler</code>	212
<code>vérification</code>	541	<code>olive</code>	201
<code>déshydraté</code>	484	<code>fruit</code>	189
<code>luzerne</code>	444	<code>prune</code>	181
<code>olive</code>			
<code>huile</code>	908	<code>légumes</code>	203
<code>agence</code>	693	<code>sécher</code>	202
<code>mélange</code>	564	<code>foins</code>	201
<code>empaqueter</code>	544	<code>fourrage</code>	199
<code>vérification</code>	496	<code>tomate</code>	189
<code>unterhaltsgeld</code>			
<code>nebenbestimmungen</code>	818	<code>übernachtungskosten</code>	600
<code>zi</code>	818	<code>erkl</code>	534
<code>louer</code>	750	<code>zuwendungsempfänger</code>	395
<code>malversation</code>	714	<code>thuringen</code>	357
<code>pénal</code>	694	<code>vor-ort-kontrolle</code>	345

Figure 3 Part of a 5-closest-neighbour list for the keywords. The first column of each keyword was calculated analogously with that of the documents in Figure 1. The second column lists keywords having a similar set of column-1-keywords as that of the keyword in focus, but they have this while never or seldom co-occurring with the keyword in focus itself.

The usefulness of creating lists of the type shown in Figure 3 is that it facilitates checking the appropriateness of the set of keywords used, whether manually or automatically assigned. Seeing the result in Figure 3, one might even find inspiration to order the keywords hierarchically (building a thesaurus).

3.1.3 Displaying ‘orthogonal characteristics’ for list items and hyperlinking

Consider again Figure 1. In addition to the closest *document* neighbours, each document could be assigned another ranked list containing its most characteristic *keywords*. Since documents and keywords in our textual data are ordered orthogonally (they correspond to the table rows and columns respectively), we can call the keywords ‘orthogonal characteristics’ of the documents, and the other way around: the documents are ‘orthogonal characteristics’ of the keywords. The same can be reasoned for Figure 3 where we could think of a third column listing those five documents where the keyword in question plays an important role. Note that the sheer absolute frequency is normally not used as an indicator of this importance, but some normalized and weighted elaboration of it is preferred. As we now are getting used to swapping object/parameter and parameter/object perspectives, let us introduce *item* as a general term referring to whatever is focussed of the parameters or objects. The facility of adding the respective ‘orthogonal characteristics’ of the items in Figures 1 and 3 has not been implemented for

these lists but it has for the dendrograms, dealt with in the next section. Leaving this section we just note that e.g. adding documents to the lists in Figure 3, these document identifiers could be activated as links whereby the explorer could click on the name to see the full-text version of the documents in which that keyword plays such an essential role and see exactly how that word was used there.

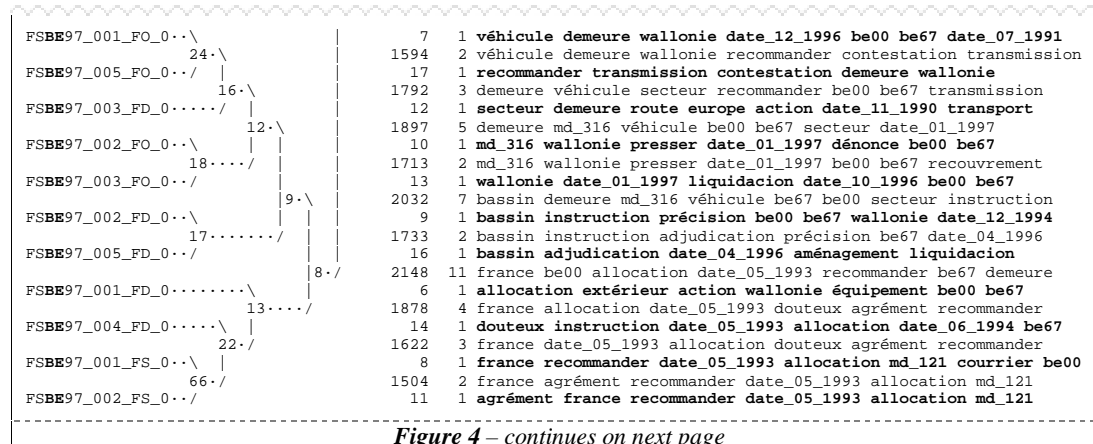
3.2 Dendrograms or ‘tree diagrams’

Advancing one step in the clustering procedure, the CA proper starts. The probably most commonly used clustering algorithm⁵ builds a tree diagram, or *dendrogram*, starting from the leaves (each representing one item in the itemXitem matrix) which it combines into twigs, branches, and then it stops when all items (in our case: all documents or all keywords) are collected as sprung out from the same trunk or *root* of the tree. This procedure has been described more in detail in a previous report⁶. The result of making such a tree based on the documentXdocument matrix in our example is illustrated by Figure 4. Building the dendrogram on the keywordXkeyword matrix, instead, will result in what is shown in Figure 5.

3.2.1 Displaying ‘orthogonal characteristics’ in dendrograms and hyperlinking

The interpretation of the dendrograms in Figures 4 and 5 has also been explained in a previous report⁶. The new facility of displaying these ‘orthogonal characteristics’ was however yet not implemented at the time that report was written and therefore we show it in this report. In Figure 4, to the right of the document dendrogram, the most salient keywords are listed in ranking order (as many as the row length allows) for each document as well as for each bundle of documents, i.e. tree nodes⁷.

In Figure 5 another idea is tested; we confine ourselves to mentioning only one specific document where a certain keyword plays the most important role. If this strongest contribution of that keyword happens to be made in more than one document, it is signalled by the ‘^’-sign followed by the number of documents in which the keyword is equally important. In Figure 5 we see, from the document names, that the keywords pertaining to this tiny part of the whole dendrogram seem to be most important in texts about Spain (as indicated by the letters ~ES~ in the document identifier), except in two cases where the texts are about France (where the document names contain the letter sequence ~FR~).



⁵ We use the hierarchical agglomerative algorithm in combination not with ‘single’ nor with ‘complete’ but with *average linkage*. This technical note about the algorithm is not indispensable for the understanding of what is reported here but it is sometimes asked for by people working with CA themselves.

⁶ In the same report mentioned in footnote number 3 above.

⁷ When two trees are combined in a node each of these new subtrees contributes to this agglomerated keyword ranking list with a weight proportional to the size of the subtree in terms of leaves/items.

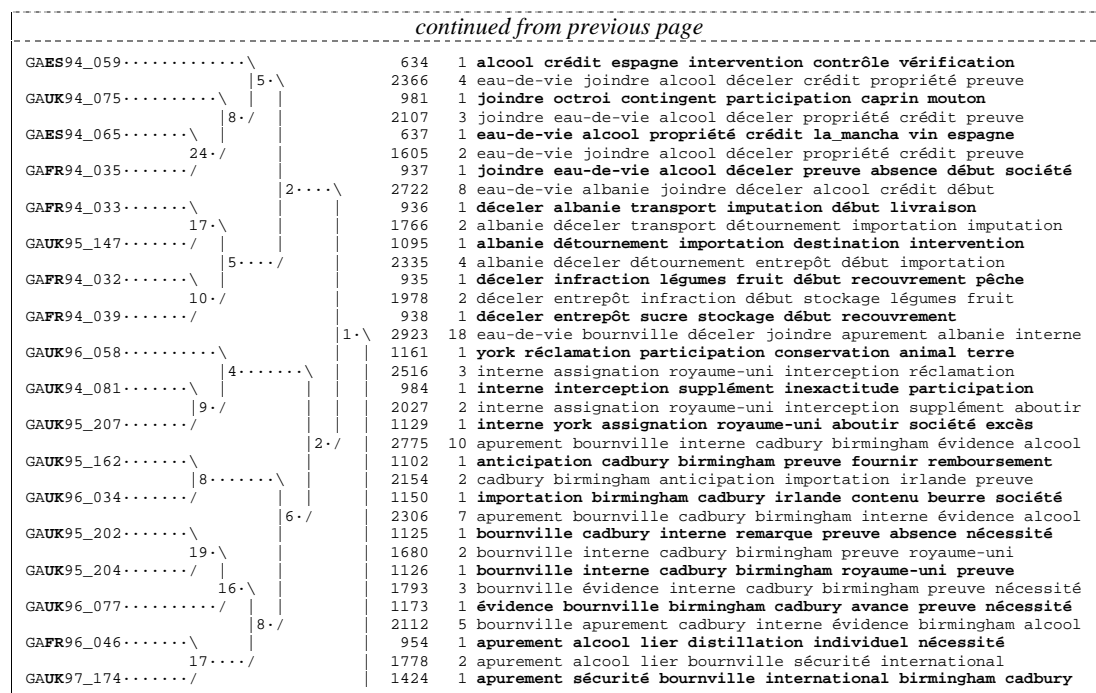


Figure 4 Two parts (together representing less than 2%) of the document dendrogram. The three columns to the right show the node number; the subtree size; and the top of the ranked list of the most important keywords describing that document (in boldface) or document group (plain text). We discover that the documents in this collection – by virtue of similar keywords – tend to group first according to investigation type (FS/GA) and then according to under which country (here: BE/ES/FR/UK) they have been filed.

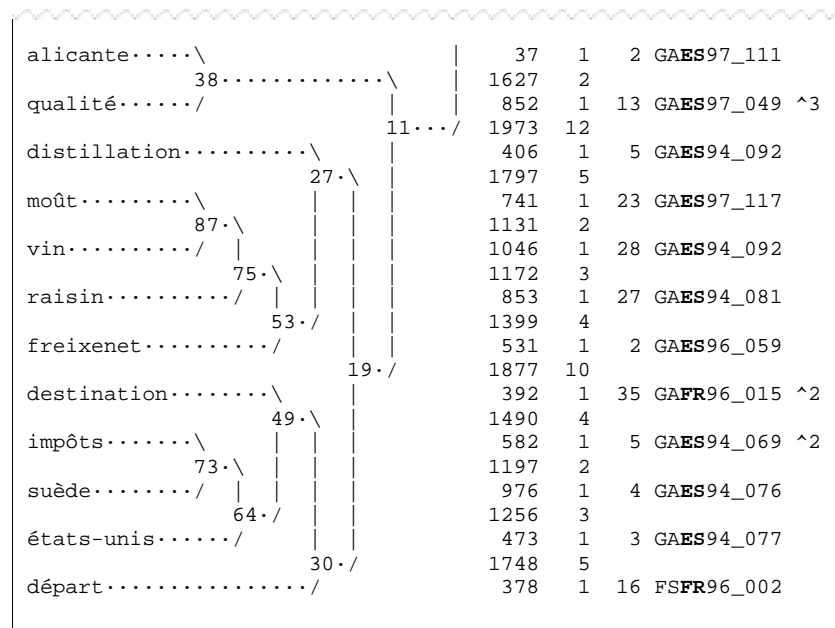


Figure 5 Part (less than 1%) of the keyword dendrogram. The five columns to the right show: node number; subtree size; number of total occurrence of that keyword; document where this keyword seems to play the most important role; and (in three cases, after a '^'-sign) in how many documents the keyword makes its most important contribution if in more than one document.

What has still not been tried out in these dendrograms are adding hyperlinks. By doing that one would be able to click on e.g. a document name to see its full text version or to see its five-closest-neighbour list. The same principle would apply to the keywords.

3.3 Item charts

A dendrogram soon becomes hard to inspect as it grows in size. One possible remedy is to chop up the tree, starting from the root, and display the subtrees in a two-dimensional *item chart*⁸ where subtrees of similar content are located in adjacent cells. We make this chart out of a square grid with an uneven number of cells at each side. The reason for an uneven number is to make it possible to have items in the *middle* of the chart. We cut up the tree in nine subtrees and order these in the chart as to optimally reflect their internal proximities given by the orthogonal data describing them. Then, in a subsequent step, we do the same thing again *inside* each of these nine cells and order even them according to internal and external neighbours. In this last split, however, not all of the first nine subtrees are big enough to be split into nine subtrees, so we must leave those unaltered. Figure 6 shows the result of making such a chart of the whole document tree, part of which we saw in Figure 4 above.

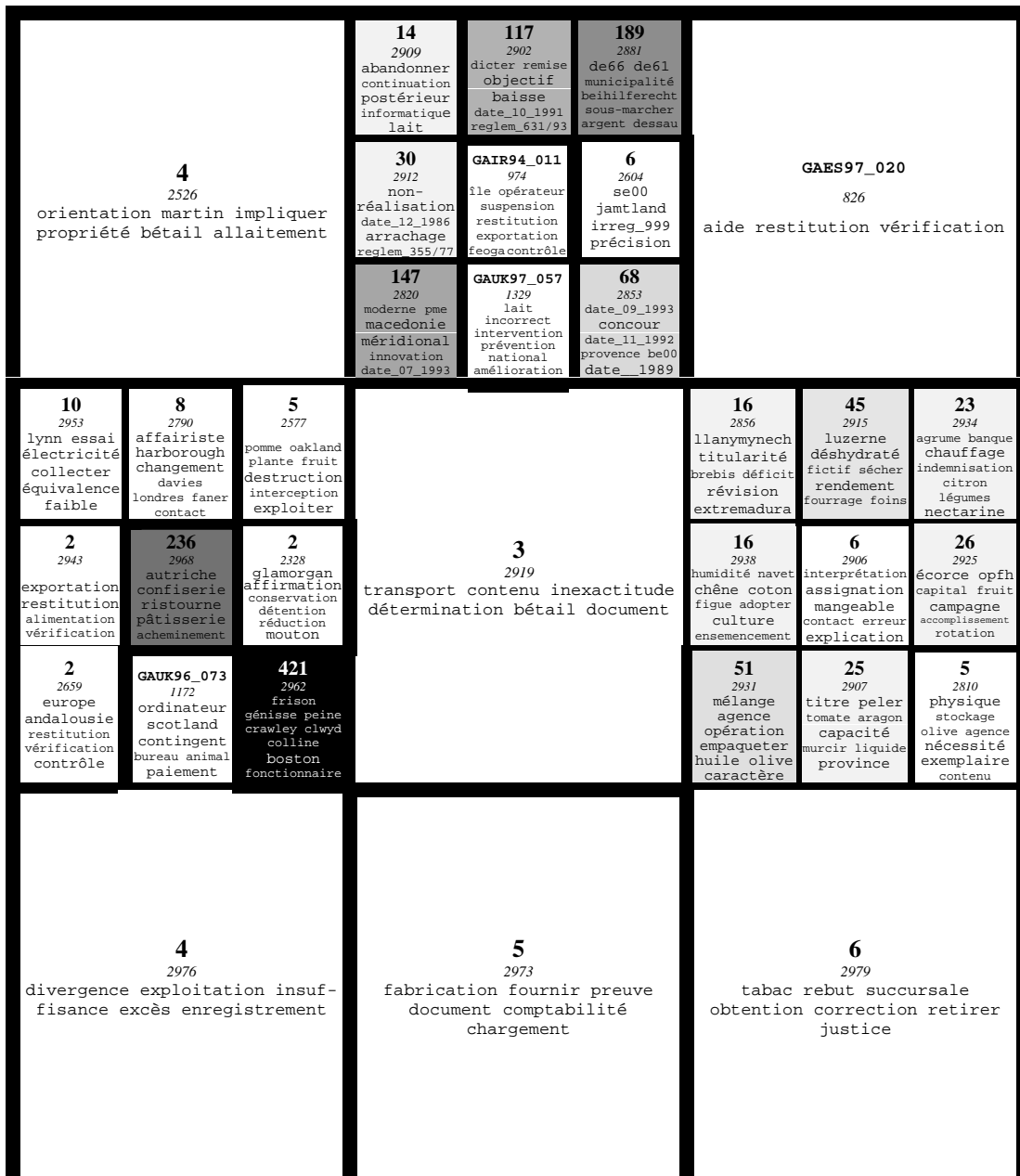


Figure 6 A tree of 1,496 documents cut up into pieces that have been more or less squeezed into the cells of a chart. The most densely populated cell contains 421 ($\approx 28\%$) of the documents, whereas cells containing only one document have this identifier written out. The cell walls reflect subtree proximity: thinner = more similar content.

⁸ Presented in J. Hagman, R. Steinberger, D. Perrotta, and A. Varfis. Tech. annex of *the 16th Int'l Conf. on A.I., IJCAI'99*, Stockholm, Sweden, 1999 and, even more in detail, in the report mentioned in footnote no. 3, above. The term *map* was used before but we now suggest *chart* as a more accurate term.

The underlying idea of creating this chart based on $(3 \times 3) \times (3 \times 3)$ grids is that we assume that a multitude is easier to “divide and conquer” if divided iteratively in three parts – easier, that is, than dividing it in, let us say, 7×7 or 11×11 cells right away, not to mention a rectangular formation. The Kohonen maps⁹, calculated by neural networks, also allow the user to decide on the number of cells in which to map the data but the data may there distribute in such a way that the original chart/map form (square or rectangular) is not reflected any more by the smaller cell groups inside it once this data→cell projection is done; often these cell groups form irregular polygons¹⁰. One advantage of having table cells formed like in Figure 6 is that such a visualization could be created automatically in a simple HTML format and thus be viewed and explored using any browser, and there, by clicking on one cell/subtree, another $(3 \times 3) \times (3 \times 3)$ chart would pop up allowing the viewer to zoom down yet another step. The document identifiers would be activated as links leading to where that document is found in the proximity list, in the document tree, or to its full-text version. Already standard HTML colour and text/table formatting commands offer useful facilities for data visualization which we have used before¹¹ with good result and adding links would be the natural next step. We also see great possibilities of making this type of visualization nicely interactive by applying JAVA scripts to such an HTML-based visualization of CA.

One of the main points of spreading out the data on some kind of chart is to better utilize a two-dimensional display area. It may be perceived a problem, however, if too much information is concentrated in too small a space (which is the case of some cells in Figure 6). Two solutions to this come to mind: **either** making the cells clickable and provide a real-time iterative 3×3 subdivision of big-enough cells in combination with some zooming facility as just described; **or** applying *controlled partitional clustering* where the user/programmer decides the number of clusters wanted and their preferred size – in this case we could tell the CA program already before it starts to calculate 81 ($(3 \times 3) \times (3 \times 3)$) clusters and then position them optimally in the chart, putting some 18 or 19 ($=1,496/81$) documents in each cell. The best idea would probably be to combine partitional clustering and “clickability”. In such a chart of 81 cells evenly “populated” on the first level, on the second zoom level each cell would contain only one or two document names ($=1,496/9/9$), activated as hyperlinks. These possibilities are “within reach” but yet not implemented by our research group at JRC.

3.4 Integrating other visualization software

CA per se works well independently of visualization modules. The latter are needed by humans to interpret and assess the outcome of the CA. The cluster analyzer developed at JRC prints its results (dendrograms and chart positions) onto simple, easily modifiable text files, which could be exported to other, commercially available visualization tools. In fact, this solution might be a very good one and that for two reasons: we are very precise about what the linguistic pre-processing and clustering algorithm should be like in order to meet our special requirements (it seems to be difficult to find anything suitable on the market) so we do not mind continuing doing that part ourselves; the second reason is that there are a few nice visualization tools around which may save us some time and which we could utilize for the presentation of the CA result. Below we will outline three ideas we find promising and where some steps have been taken already to adapt the CA output to be inserted in available visualization tools. The subsection on ThemeScape is a reshaping of part of a previous report¹².

3.4.1 *The Link Notebook™ by i2 Limited*

i2 Limited, UK, offers a software tool for creating link charts. It is called *Link Notebook™*. We see the possibility of using this tool to visualize *some aspects* of our document and keyword clusters. The tool accepts input in plain text describing these items and what relations hold between them (something which will be provided by our cluster analyzer) and then automatically draws up a map where each item is represented by an icon and all relations to other items are displayed by means of labelled links. Both icon images and relation measures and thresholds are user-defined and the program iteratively optimizes the layout of these maps to facilitate its interpretation. Our research group is already working with software from the same company, the *Analyst's Notebook™*, and we understand the OLAF staff has experience with this software as well. As soon as time permits we will start porting some CA output into this tool to see what could be accomplished.

⁹ See e.g. T. Kohonen. *Self-Organising Maps*. Springer, Berlin, 1995.

¹⁰ This can be seen at e.g. http://www.telegeography.com/Resources/md_fl_4.html.

¹¹ See Figures 11-16 in J. Hagman (2000) *Construction and Performance of a Language Recognizer*, Modus Operandi deliverable 8, carried out by the JRC for the European Anti-Fraud Office OLAF.

¹² R. Steinberger & J. Hagman (2000) *Commercial Keyword Identification and Clustering Software*, Modus Operandi deliverable 10, carried out by the JRC for the European Anti-Fraud Office OLAF.

3.4.2 GIS systems

Within our JRC research group we also do work using GIS (geographical information systems). Such a system connects relational databases to “ordinary” geographical maps where it visualizes, e.g. by colouring dots for cities and polygons for regions or countries, what geographical parts are related to the data. The data could be about anything, provided it has some relevance to the map of interest. The data used in the experiment illustrating this report frequently refers to European place names or “areas of particular interest” in the context of ongoing or planned fraud-related investigations. By recognizing geographical names in texts, any recurring theme in a document collection, e.g. *cigarette smuggling* or *faulty olive oil quality labelling* within the EU could be visualized on such a GIS map if connected to our other modules. This would not even require any CA to be carried out; just the linguistic pre-processing, information extraction, name recognition, porting and, finally, the GIS display of the relevant areas mentioned in those documents. We hope to find some time to explore this yet unleashed potential of connecting automatically extracted geographical information in texts with GIS.

3.4.3 ThemeScope

NewsMaps.com is a web service¹³, provided and operated by *Cartia, Inc.*¹⁴. On a regular basis, web pages are scanned, linguistically analyzed and clustered according to their most salient semantic content descriptors. This CA is then projected onto two-dimensional maps where more similar news stories come closer by virtue of their similar vocabulary. In regions of the map with high density of news (individually represented by dots, as small villages), the whole landscape is raised to visualize the accumulation of texts dealing with that subject. On the other hand, news reported with rare vocabulary with respect to the whole agglomerated article collection are represented as located in isolated areas, and the lower the density, the lower the altitude on the map¹⁵. Where no news at all are present on the map, including its entire periphery, there is water. The maps are interactive, i.e. the user can zoom into interesting parts (or have them indicated by means of a search mechanism) and there see what themes and articles there are to read. By then clicking on a “news item village”, the reader is linked to the original site and can there read the article exactly the way it was intended by its author.

The clustering and visualization software making all this possible is called *ThemeScope*TM and is a state-of-the-art generic tool, but as we would like to use our own specialized and – as it seems to us when reading their product description – linguistically more precise modules, we would prefer doing the first steps ourselves and then porting the outcome of our CA to *ThemeScope*’s visualization module. In recent discussions with a representative of *Cartia Inc.*, we have been informed that a new version will be issued within some three months from when this report is written and we intend to investigate this possibility further, based on what that version offers concerning portability, price, and performance.

4. Summary and Conclusions

At an early stage of the Modus Operandi project, the JRC project team opted for building *in-house* a stand-alone parametrizable cluster analyzer, specifically suited to the needs of the project. The alternative would have been to spend time on searching the market for suitable commercial tools – with no guarantees for success – going through a lengthy purchasing procedure, installing and tuning the software, before actually setting to do any clustering. We were thus constrained to give priority to the cluster analysis task per se, more than to the subsequent visualization of cluster analysis results. Nevertheless, we think that the basic visualization facilities developed for the purpose are satisfactory.

The modular cluster analyzer developed can export intermediate results into an ordinary text format. It is therefore feasible to consider integration with other commercial visualization tools to present the results of the cluster analysis program. Powerful, user-friendly data visualization is a very lively research area at the moment and software providers are coming out continuously with new and better products.

As far as the follow-up to this project is concerned, in the context of separate institutional research efforts, it is our team’s intention to assess some commercial visualization alternatives, such as the ones mentioned above, namely, the Link Notebook, GIS systems and *ThemeScope*.

¹³ At <http://www.newsmaps.com>.

¹⁴ At <http://www.cartia.com/products/index.html>.

¹⁵ A similar cluster visualization method was presented in J. Hagman, *Molecules and Mergers: Towards a More Concrete Appearance of Cluster Configurations and Scattergrams*, Proc. IASTED Int’l Conf. on Computer Graphics and Imaging, (ed. M. H. Hamza), Halifax, Canada, 1998, (pp. 166-170).