

An Implemented Cluster Analyzer for Documents and their Indexing Terms

Deliverable 12a of the *Modus Operandi* project
carried out by the JRC for the
European Anti-Fraud Office OLAF

Administrative arrangement 14408-98-10

November 1999

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>

An Implemented Cluster Analyzer for Documents and their Indexing Terms

1	Introduction	1
2	Implementation and characteristics of a cluster analyzer (CA)	1
2.1	Choice of programming language and environment	1
2.2	The input data: its format, loading, and representation inside the CA	1
2.2.1	The list of indexing terms to be <i>ignored</i>	1
2.2.2	The list of indexing terms to be <i>considered</i>	2
2.2.3	The document profiles	2
2.3	Minimum-occurrence check of indexing terms	3
2.4	Assignment of 'document-representative values' to indexing terms	3
2.5	Calculation of similarity matrices and dendrograms for <i>documents</i>	6
2.6	Calculation of similarity matrices and dendrograms for <i>indexing terms</i>	7
2.7	Cellular projections of the dendrograms and indexing terms	10
3	Current state and usage of the CA	13

1 Introduction

Current work and study on cluster analysis is carried out within the *Modus Operandi* project. From the JRC ISIS Annual Report of 1998, we make the following extractions as to give an idea of the context:

Modus Operandi [has] the objective of applying Language Engineering ... to support [OLAF] staff by automatically analysing and processing large amounts of textual information written in any of the European languages. [OLAF]'s needs include finding documents relevant to current interests, assessing the contents of documents quickly, extracting relevant bits of information, presenting them in an informative manner and, possibly, deriving meta-knowledge from texts such as ways in which fraud develops over time and which countries become involved with specific products. The main goal ... is to develop a prototype system which automatically ... identifies ... keywords ... detects the subject domains [of texts] and groups them according to their similarity with each other. ... The motivation for the grouping is to see whether the clusters of fraud-related texts can be used to derive a fraud case classification. Techniques will be investigated which allow retrieving relevant documents from multilingual ... collections such as the internet or [OLAF]'s intranet. Further activities may include ... conceptual ... document indexing.

[Barbas & Steinberger 99, p 25]

Cluster analysis is called upon in the 'grouping' here, and is also underlies information presentation.

2 Implementation and characteristics of this cluster analyzer

In this second section, constituting practically the whole of this report, we will go through the main modules of the cluster analyzer (CA from now on). We will see the input and output formats of the files involved in the process and, above all, we will discuss some linguistic and computational issues of crucial importance to the final result of the CA. This report will thus try to lead the reader through parallel reasoning lines, bringing up, at the relevant CA module, the most important observations as we proceed along its algorithm and that of the visualizer extension outlined in the following subsections.

2.1 Choice of programming language and environment

The computationally heavy cluster analysis, characterized by thousands of operations on floating point values, suggested choosing a sequential, imperative, widely known programming language, with compiler and executables being up-to-date and of excellent performance; the choice was the C programming language. Just ordinary ANSI C has been used, without any C++ objects. On recommendation from professional programmers the development environment *CODEWARRIOR Professional (Release 4)* was purchased and used for the implementation. Our choice warrants broad compatibility with other C libraries and even other programming languages, such as PASCAL and JAVA. CODEWARRIOR is platform-independent although the programming has been carried out within the Windows NT environment.

2.2 The input data: its format, loading, and re-structuring inside the CA

The CA takes three data sets as its input, each consisting of a structured list in a file of normal ASCII text format. At present there must be a first line in each file which indicates how many items (i.e. stop-words / indexing terms¹ + inflected variants / document profiles) the list contains. The reason is that this allows the CA to allocate immediately the memory required for storing the data in RAM and then transfer it there as it scans through each file only once. No item may contain any ' ', i.e. blank character, as that would be interpreted as two terms separated by ' '.

2.2.1 The list of indexing terms to be ignored

The automatic assignment of indexing terms to documents is done by another application² (and the principle of this procedure is briefly outlined in 2.4 below) but sometimes not all of those proposed indexing terms are found useful for cluster analysis. Reasons for this are e.g. that they belong to a non-wanted, maybe technical notation or to another language, or that their frequency or document-specificity is too low or too high to be of any interest as semantic indicators of the content of that document as it will be compared to those of other documents in the analysis. Therefore a list is created (semi)manually which works as a second-step 'stopword list', following the first-step list involved in the preceding calculation of document profiles. This list has a simple format as shown by Figure 1:

¹ The terminology varies between e.g. 'keyword', 'index(ing) term', and 'descriptor' -- here all used as synonyms.

² I.e. by a customized part of the program WordSmith™ by Mike Scott, Liverpool University, UK [Scott 99].

```
8477
aan
ab
above
ac
acht
acima
acqui
ad
...
```

Figure 1 List of terms to ignore.

This list just starts with a number of how many words there are in the list and the list proper follows right after (i.e. no blank line) with one word per line. In the code for the CA, for the reader who is reading that in parallel to this report, each ‘stopword’ (as the term is used here) is stored as a `character string` in a C-structure called ‘STPWRD’ along with information e.g. about how many documents that stopword describes. This information is useful for statistical analysis relating to the corpora involved in the preceding indexing term assignment, which one might want to revise.

2.2.2 The list of indexing terms to be considered

The list of indexing terms to be considered by the CA has the same format as that of those to be ignored but with the additional possibility to associate to each indexing term a sublist of “variants” of this term. The user would typically consider as variants inflected forms of this “head term” (preferably expressed in some basic form, as a noun in singular or a verb in infinitive) but this sublist may also contain synonyms of the head term, or otherwise associated (semantically close or thesaurus-suggested) terms – even terms belonging to another language than the head term; all user’s choice. Traditionally lexicographers and other linguists refer to the reduction of inflected word forms (all originating from *one* and only *one* basic word in *one* language, though) to their basic form as *lemmatization*. This however does not mean that the definition of ‘lemma’ is fully agreed upon among professionals. For our purpose here we thus definitely allow a wider association of wordforms to one basic form and apparently we also use ‘lemmatization’ in a wider sense. Figure 2 shows the beginning of a file of association to chosen basic lemmas:

```
5919
abandonner
  >abandonnee
  >abandonno
  >abandonou
  >abandonare
abbattre
  >abattage
  >abattent
  >abattoirs
...
```

Figure 2 List of terms to consider.

The format of an optional sublist of this kind expects the subterms to be listed after the head term, each indented by one tabulator and preceded by the character ‘>’ (i.e. ‘greater than’) with no blank between this character and the subterm. The number of the first line refers to the total number of wordforms, i.e. head terms plus subterms. The subterms will be used for *matching* when reading the document profiles, but the header terms will be the only ones *counted* and *considered* in the subsequent cluster analysis. If possible, this ‘subterm → head term’ association should be done during the preceding keyword identification but this is not always the case. The C-struct representing each indexing term contains, besides the `string` of the term itself, information on e.g. whether or not it is a subterm and, if so, what is its head term and how often it is used as a document descriptor.

2.2.3 The document profiles

There are two formats by which the document profiles can be given. They both start with a number of how many documents are described (each with a set of indexing terms) in the file. The simpler format has the document identifier starting each line followed by a tabulator and then a list of ‘ ’-separated indexing words without any other information. This format is shown by Figure 3.

```
260
DocA001 . access_information jrc institute environment research commission activity external
DocA002 . access_to_information commission access public document principle
DocA003 . action_for_annulment refund export unduly commission pay ecu gradin member amount
...
```

Figure 3 Top of the simpler file of document profiles having only (implicit) binary occurrence values.

The more elaborate format, shown by Figure 4, expects the first-line number to be preceded by a ‘-’, i.e. a minus sign and then each document identifier starts on a new line (exactly two blank lines after a

preceding document profile) and its profile follows in terms of triples, each on a new line; consisting of an indexing term and two numbers referring to the term's "representativeness" of that document.

-1600		
Doc#0001		
irrégularités	8	232127
demande	5	133704
dedans	4	101103
rédigé	4	101103
factures	4	101103
renovation	4	101103
suggestion	4	101103
arrivée	4	101103
envoyés	3	68777
Doc#0002		
lettre	10	308683
helléniques	9	274575
jeunesse	7	206440
dénonciateur	6	172440
bénéficiaire	6	172440
vérifier	6	172440
dépenses	6	172440
bénéficiaires	5	138522
enquête	5	138522
...		

Figure 4 Top of the more complex file of document profiles with frequencies and keyness values

The current version of the CA does not require separate file extensions for these three types of input data; it simply expects them to be in simple text format with whatever name. It is nevertheless advisable to use different file suffixes (e.g. `.stw/.stp`; `.lex/.kwd`; and `.dcs/.dat` respectively) for practical reasons, to distinguish them better from each other as one's library of files easily grows.

2.3 Minimum-occurrence check of indexing terms

When the document profiles are loaded into the CA, each indexing term is checked whether it should be ignored (according to the 'stopword list', or, shorter still: 'stoplist') or whether it should be counted as it is or as a subform of another term (according to the previously loaded list of 'indexing terms to be considered' – 'indexlist' for short). Having done this, all terms pass through a check that (1) each term encountered is matched by and "taken care of" either by the stoplist or by the indexlist; and (2) if they are on the latter list, that its head term occurs as descriptors for at least N documents in the collection. The value of N is set by the user and is of course ≥ 2 (documents) because with occurrence of only 1 that word would not bind any two documents to each other in the cluster analysis and would therefore be useless for that purpose. There is also a third case (3) where the encountered term is not foreseen by any of these lists. Depending on different settings of the program, a term which in this way falls outside both lists could be dumped to a file for later examination and completion of these lists. Note that this procedure makes both inclusion and exclusion of indexing terms *explicit*. However smart and efficient the clustering algorithm, it can never make up for sloppily ordered or trashy input data. The quality of the CA output is rather a *product* than a *sum*, and we know what happens with the product if even only one out of several potent factors is zero (e.g. the input quality). During this check, the approved index terms in the list, which has actually been a temporary one up to that point, are passed over to a definite list. In C jargon we express the same thing as all `INDEX structs` in the array `TmpIdxArr` having a value of `NoOfDocs \geq MinValOcc` are passed over to the final array `IdxArr` for further use.

2.4 Assignment of 'document-representative values' to indexing terms

How much an indexing term contributes to *represent* the content of a particular document is a less trivial issue in automatic document categorization and cluster analysis. Before treated by the CA, the keywords representing texts, or, in other words: 'the set of indexing terms constituting the document profiles', have been chosen on a statistical basis comparing their relative occurrence in a given text to that of the same term in a reference corpus, i.e. a collection of "normal" texts which together are supposed to be "representative" of the language in question. What is to be considered such a "normal corpus" is, by the way, a linguistic research topic in its own. Terms in the examined text which are relatively more common than they are in the chosen reference corpus are listed, often in ranking order, as automatically derived keywords for that text. The relatively more frequent a term is in a text with respect to the reference corpus, the higher a *keyness value* it will be assigned and this value could also be calculated according to different formulæ. The user then decides on a lower threshold of keyness value when creating keyword profiles for a collection of documents. Of course, the lower this threshold is set,

the richer the information will be which underlies the comparison of the documents in the clustering phase. But, on the other hand, with too low a threshold, the more uninteresting keywords will enter the game, bog down the calculations and make the result ungainly and cluttered with irrelevant curiosities.

As described above in 2.2.3, the CA receives groups of already calculated indexing terms along with their absolute frequency in the document and the corresponding keyness value. Let Table 1 illustrate a very simple example of three documents with their respective profiles (terms ordered alphabetically, F = frequency, K = keyness value):

Doc#1			Doc#2			Doc#3			Σ Docs
Index term	F	K	Index term	F	K	Index term	F	K	F
falsification	3	93.5	investigation	4	81.0	falsification	3	75.0	23
investigation	7	54.1	lens	1	106.1	investigation	2	12.4	201
lenses	2	101.3	lenses	1	132.7	lenses	3	122.0	2
			sunglasses	1	260.9	sunglasses	2	159.8	13
									3

Table 1 The first three out of several document profiles characterizing the content of a collection of documents.

Intuitively the keyness value is preferable since it better indicates the relative importance of that term as a document descriptor than does the plain, absolute frequency. For the same reason it is also preferred in the cluster analysis. This is also the normal procedure but in this example we note that we have both `lens` and `lenses` present in Doc#2. In subsection 2.2.2 above we mentioned that the ideal case is to have both the document collection and the reference corpus *lemmatized* before statistical comparisons are being done, and we also said that in lemmatization each word is made refer to what is considered its main *lemma* or ‘headterm’. Working with lemmas only, it all becomes a whole lot more manageable (since the number of word forms is largely reduced, especially in more inflectional languages than English – which most other EU languages are). It is also more accurately representative of the document content since inflected forms of the same lemma would otherwise not be recognized as having the same “basic” meaning, which is the case of `lens` and `lenses` in this example.

Back to our example: let’s say we want to capture the (quite obvious) semantic similarity between `lens` and `lenses` at this late stage. The problem is that the more rarely occurring form `lenses` will have a greater impact on the keyness value than more commonly occurring form `lens` and this is *not* what we are interested in measuring. In fact, it actually interferes undesirably when calculating document similarity based on shared indexing terms (and the co-occurrence of indexing terms themselves – more about this later). So what do we do with the K-value? The “craftsman’s judgment” – the craftsman being the computational linguist since both statistical/computational *and* linguistic effects must be taken into account – suggests that in these unhappy and hopefully rare cases we apply an alternative procedure and algorithm, i.e. we pick the plain frequency values for each subterm and add them to the frequency of the head term. We will thus not be able to use the K-values anymore but we will present a remedial method in what follows. The profiles would first be re-calculated to what we see in Table 2:

Doc#1		Doc#2		Doc#3		Σ Docs
Index term	F	Index term	F	Index term	F	F
falsification	3	investigation	4	falsification	3	23
investigation	7	lens	2	investigation	2	201
lens	2	lens	2	lens	3	15
↑	↑	↑	↑	↑	↑	↑
		sunglasses	1	sunglasses	2	3

Table 2 Same as Table 1 but with all forms of `lens` merged into one form and the frequencies duly adjusted.

The K-values (which we cannot use any more) gave us the *relative* importance of the terms to a given document. That information must be considered richer than the simple absolute frequency value we are left with at this point. Confronted with our little dilemma before, we have traded the K-values for lemma-based index term structuring, probably a good choice when the best, ideal situation is not the actual one. Now, we could proceed by simply using these raw frequencies when calculating the similarity between documents but the effect would then be that those indexing terms which dominate the document profiles quite brutishly by higher frequencies will also have a stronger influence on the future cluster analysis. Probably this is not what we want since that frequency difference may very well reflect the ordinary frequency distribution in that language and thereby would the interesting document-specific variations manifested by other, less frequent terms “drown” in the context. The question can be posed with another example: if we have a one-page text about lenses, would we rather consider that text more similar to another document also consisting of one page which has proportionally quite a lot of words relating to lenses *or* would we prefer consider this first document more similar to, say, a thick catalogue which in absolute numbers has more lens-related terms in common with that document

but in relative numbers makes a much lower degree of match? The choice, if abstracted further, is between maximal *total* vs. *partial* similarity and, again, it is the user's choice and will probably depend on the purpose of the cluster analysis; would we like to have the catalogue in our example swallow smaller documents which it is partially similar to or do we prefer to have small, in their entirety similar documents cluster more closely and the big catalogue kept further apart?

As mentioned, there are ways to make up for the lost K-values to some extent by calculating new relative values, but of another kind, denoting another relativity. In fact, at least two alternatives come to mind quite intuitively, viz. either to relate each index term to its "team mates" which describe the same document or to itself as it is used in describing different documents throughout the collection. We will see the effects of both these alternatives, starting with the first. In Table 3 we calculate the relative frequency contribution (%F) of each term inside a single document.

Doc#1			Doc#2			Doc#3		
Index term	F	↓%F↓	Index term	F	↓%F↓	Index term	F	↓%F↓
falsification	3	25				falsification	3	30
investigation	7	58	investigation	4	57	investigation	2	20
lens	2	17	lens	2	29	lens	3	30
			sunglasses	1	14	sunglasses	2	20
Σ	12	100	Σ	7	100	Σ	10	100

Table 3 The K-values of Table 1, cancelled in Table 2, are now substituted by the %F-values.

Note that basing the cluster analysis on the %F values instead, each document will have an identically large influence since they sum up to 100% for every document. This is not the case when using the K-values; documents there with exceptionally high or plenty of high-scoring K-values have a bigger impact on the clustering tendencies. Well, making each document equally "heavy" like this may be desirable in some applications – the one-page document in our aforementioned example would here be as influential as the whole catalogue, for instance. Again, it all depends on the user's preferences.

In Table 3 there is however still an effect of "brute" frequency influence as highly frequent terms (as *investigation*) quite mercilessly dominate less frequent terms (as *sunglasses*) although the latter might be at least as interesting when associating documents to each other. The other way of calculating relative values does not relate the frequency of a given index term to the sum frequency of *other* terms describing the *same* document but to the sum frequency of the *same* term describing the *other* documents. Doing this, another phenomenon of 'equal impact' arises, not relating to documents but to indexing terms as each one of them will have their 100% distributed over all the documents. We see the outcome of this way of relating in Table 4. Now again the contribution of each term inside a single document varies, as it did using K-values. Documents holding many less commonly occurring index terms will contribute more strongly to the forces deciding the cluster analysis and this is often wanted.

Doc#1			Doc#2			Doc#3			Σ Docs
Index term	F	←%F→	Index term	F	←%F→	Index term	F	←%F→	F
falsification	3	13				falsification	3	13	23
investigation	7	3	investigation	4	2	investigation	2	1	201
lens	2	13	lens	2	13	lens	3	20	15
			sunglasses	1	33	sunglasses	2	67	3

Table 4 The %F in Tab. 3 are based on Σ F of each document; those here are based on Σ F of all documents.

The way of calculating relative values shown by Table 4 is probably the better one to compensate the lost K-values. The main difference between these two types of relative values is that the %F-values of Table 4 (1) guarantees each indexing term to have an equal overall influence on the clustering, and (2) that as the K-values express the "importance" of the terms to the documents in relation to an "external" reference corpus, the %F-values use the document collection *itself* as the corpus. So (and this applies to *both* the unfortunate case where we are not able to combine K-values and lemmatization *and* where we are luckily able to do that) an adequate procedure could be to create firstly a frequency list of all occurrences of calculated index terms in the collection in order to see what vocabulary characterizes our collection with respect to the reference corpus, and, secondly, to recalculate the data as in Table 4 to see how each document relates to the set of documents of which it is an element.

Before closing this section: there is also the possibility of assigning a *weight* to each term which may be statistically given or not. In the latter case, we could think of the user of a search system who more or less manually assigns weights of importance to the words of a query. In the former case one could consider terms typically correlating strongly with other terms as more "stable" and thus trust them with a stronger indexing capacity – or the other way around: arguing that strongly correlating index terms, at least those correlating with a value very close to 1.0 (i.e. maximum), ought to be grouped together with

the terms they correlate so strongly with and thus instead be assigned a lower weight. At the moment this report is written we have not yet had the time to experiment with weighting based on correlation but there are a few ideas we would like to try out in the future. Note, also, that the whole reasoning here concerning the frequency values of the indexing terms would very well be applicable to keyness values, whereby we would expect a combination of advantages.

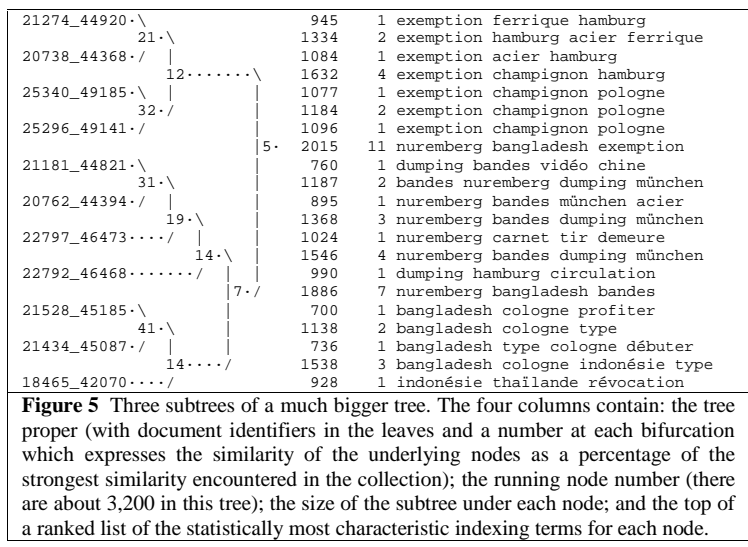
2.5 Calculation of similarity matrices and dendrograms for documents

At this step of the CA program a similarity index is calculated between each pair of the whole document collection in terms of the indexing values one has decided to use (refer to the discussion in the above section 2.4) and their values as given by the algorithm finally chosen. In detail: for each document pair and, there, for each index term present in both documents either the lowest or the sum of these values are taken and added to the summed similarity index. E.g. the similarity index for Doc#1 and Doc#2 in Table 4 above could be $\min(3, 2) + \min(13, 13) = 15$ or $\text{sum}(3, 2) + \text{sum}(13, 13) = 31$. This sum is then written into a document matrix containing the similarities between all document pairs. The minimum similarity will, of course, equal zero, but the maximum similarity will in this way not be pre-defined; it will depend on the number of indexing terms matched and their own frequency and distribution. All similarity values can, nevertheless, be re-calculated to percentages of the maximum value found in the collection in a second round. There is also a check for documents with identical profiles (i.e. terms and values) which will alert the user of suspected duplicates in the document collection.

Based on the similarity matrix, a tree diagram is created. Figure 5, below, shows three tiny parts of such a 'dendrogram'. The method is hierarchical and agglomerative, applying *average* distance, which has been found more accurate than the *single* or the *complete* distance. Our collection in Table 4, provided there are no duplicates, will make the CA create a symmetric matrix of (1600x1600) cells, i.e. close to 1,3 million unique document pairs. This matrix is scanned for the largest similarity value and the two documents involved are combined as two leaves as a beginning of the document tree being built. If several document pairs share the highest similarity value, that pair with fewest indexing terms involved is favoured. Each document has, as noted, in its data type an array with all its indexing terms and values (same info as shown in Table 4) and when combining two documents/leaves into a node/twiglet, the two horizontal and the two vertical lines in the documentxdocument matrix is substituted by lines with average values, and the index term array in the new node contains the union of those of the leaves (corresponding to the columns in Table 4), but the values there are set to the average. This procedure is carried out iteratively and soon entire "twigs" will start being combined with document-leaves or other twigs (which eventually will grow to ever bigger branches) and then all average values are calculated based on weights so that a bigger subtree contributes proportionally more both to the new average value in the matrix and to the relevance ranking of the index terms of that subtree.

15892_39034... \	241	1	tourisme constituer analyse
15885_39027 \	1411	3	tourisme hôtel banque constituer
20... /	239	1	hôtel tourisme constituer analyse
15891_39033... /	1343	2	hôtel tourisme banque garantir
	247	1	tourisme hôtel garantir banque
9...	1781	8	area tourisme hôtel tejo banque
16216_39395 \	301	1	area exploration investir ajouter
18... \	1381	2	area exploration investir ajouter
16132_39303... /	408	1	area approuver executer
16... /	1449	5	area tejo hôtel tourisme
15888_39030 \	250	1	area hôtel tourisme effective
23 \	1296	2	area hôtel tourisme effective
16664_39866... /	309	1	area éducation effective licence
19... /	1373	3	area tejo hôtel tourisme
15904_39047... /	206	1	area tejo retirer lisbonne projet
16578_39780 \	497	1	carabinieri puglia bari anonyme
28 \	1209	2	carabinieri puglia olive bari
29353_54661... /	675	1	carabinieri olive puglia
20... \	1339	3	carabinieri salerno puglia tâche
29377_54705... /	916	1	carabinieri salerno transformer
7... \	1896	6	carabinieri puglia interpreter
28847_53729... \	1038	1	russie détriment avocat puglia
9... /	1788	3	puglia interpreter russie
2891_5745... \	205	1	puglia présumer éléver interrompu
17... /	1432	2	puglia interpreter route présumer
28852_53735... /	236	1	puglia interpreter route taxe
4...	2053	12	carabinieri liguria puglia olive
29357_54669 \	576	1	olive agence huile tâche débiter
22 \	1312	2	olive agence huile tâche stockage
29358_54670... /	913	1	olive agence produit
12... \	1642	3	olive huile agence tâche stockage
5374_10136... /	948	1	olive difference liquide huile
9... /	1808	6	liguria olive huile stockage
28898_53793... \	174	1	liguria commune globe spécifique
25... /	1251	3	liguria commune olive huile globe
28903_53800 \	26	1	huile converser stockage liguria
25... /	1238	2	olive huile liguria stockage
16082_39249... /	89	1	olive archive liguria huile

Figure 5 <figure continues on next page>



What is the practical use of document trees? There are several. The first effect they have is that of *data visualization*. The user will see how the document collection “spontaneously” orders itself, based on the distribution of the index terms that are considered. Viewing the documents in this binary tree structure, the user can further explore the collection and compare the major branches of the tree to pre-defined categories in which the documents were grouped before being clustered, if such categories exist. This comparison could be part of an evaluation of the categorization/classification system used – how relevant it is for the real data at hand – and whether it could underlie a revision of that system. Inspecting the three parts of the dendrogram shown in Figure 5, the document names unfortunately do not say sufficiently about whether they belong to particular pre-defined categories. Knowing that these identifier names/numbers have been created and assigned sequentially category-wise before the clustering, we can nevertheless discern a certain continuity of the pre-defined categories as their documents arrange in the tree and by looking at the most prominent indexing terms to the right we get a clearer idea about how this came to be. In case there are no pre-defined categories, a cluster analysis, whether its result is presented as a dendrogram or cut up in lists (possibly still based on an underlying tree diagram), can *suggest a data-driven categorization system* which could also be *dynamic* as new data is added to the pool of documents.

2.6 Calculation of similarity matrices and dendrograms for indexing terms

Let us go back again to Table 4. When we calculate the similarity indexes for indexing terms we see how they appear together as descriptors for the same documents and we add this co-occurrence of each pair of index terms throughout the collection. Neither here do we use the brute absolute frequency values, but rather those denoted ‘%F-values’. As was the case with the inter-document similarity, we may here as well choose a corresponding $\min(13,3)+\min(13,1)=4$ or $\text{sum}(13,3)+\text{sum}(13,1)=30$ when calculating the similarity adds of the terms (*falsification* and *investigation*) for the documents Doc#1 and Doc#3 in Table 4 as a small part of a supposed bigger document collection. For yet another aspect, we refer to Table 5:

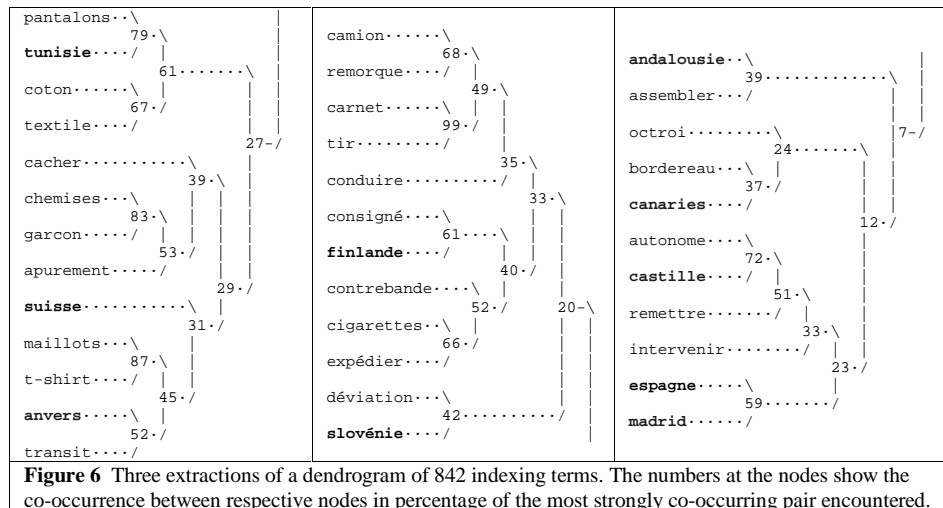
Doc#1			Doc#2			Doc#749			Σ Docs
Index term	F	←%F→	Index term	F	←%F→	Index term	F	←%F→	F
falsification	3	13	investigation	4	2	investigation	7	3	23
investigation	7	3	lens	2	13	lens	2	13	201
lens	2	13	sunglasses	1	33				15
									3

Table 5

In which document would we consider the terms (*investigation* and *lens*) most strongly “correlated”, within Doc#1 or Doc#749? Maybe the latter since there we can be sure that the high frequency, 13, of *lens* is less likely to be due to the presence of *falsification* or *sunglasses*. In Doc#749 we see that *investigation* and *lens* independently of other indexing terms together describe the document and we may want to assign some “bonus” to the similarity index within this very document for this reason; their correlation simply seems better corroborated in this observation. One way of assigning this bonus value is to play with the number of indexing terms used in total to describe a document, irrespective of their values and then let this number have some reasonably modest influence in “penalizing” longer lists of indexing terms, e.g. dividing the similarity index (however it has been calculated

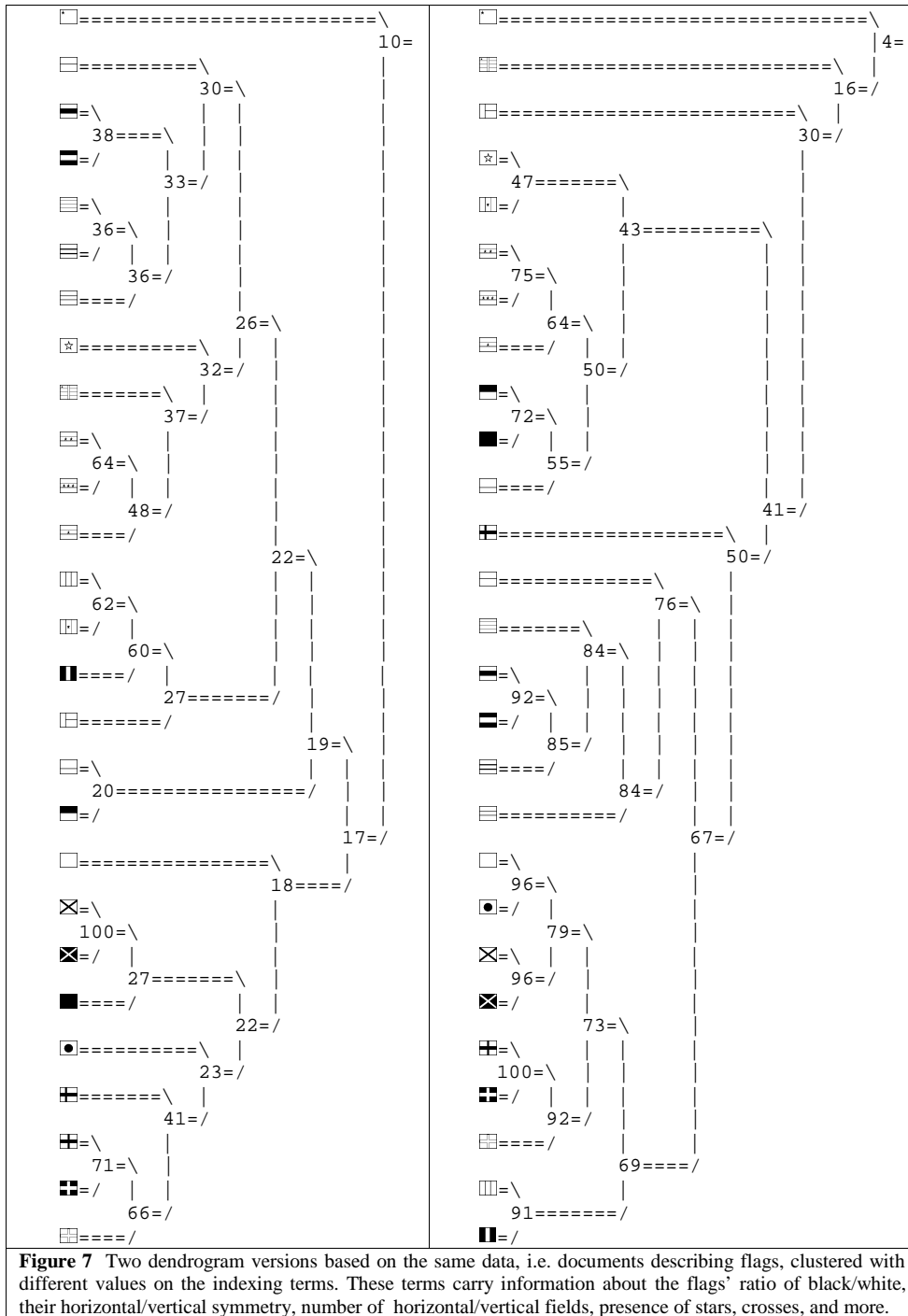
up to that point) e.g. by $(98+N)/100$ where N is the number of indexing terms describing the particular document. This way each other term not involved in the matching will impair the similarity index with 1%. This algorithm could, finally, just a little bit more elaborate, be made to automatically adjust to the maximum number of descriptors found for one document in the collection.

The creation of the similarity matrix and the dendrogram of the index terms is carried out very much the same way as that of the documents. For the similarity matrix, an appropriate algorithm is chosen, taking the above discussion in consideration, and then the tree diagram is built by iteratively searching the matrix for the highest co-occurrence indicator, given the algorithm applied. Figure 6 illustrates three pieces of a dendrogram constructed of our sample. The possibility of indicating, to the right of the tree, in which document each term gives its most significant content-descriptive contribution has been considered, but is yet not implemented. Doing that, the form would be exactly the same for both the document and the index term tree.



Recapturing the discussion above about the assignment of values to the indexing terms is based on statistics of their distribution throughout the collection, let us mention the possibility to associate to each term other importance indicators. If the user wishes to enhance the geographical data, for instance, we could think of special lists assigning extra weight to the boldface keywords in Figure 6. This is an interesting possibility as it makes it possible to connect pre-categorized lexical forces to the automatically derived forces underlying the cluster analysis. By ‘pre-categorized lexical forces’ we intend structured lists of e.g. names of geographical, personal, company or commercial entities. As a matter of fact, the CA can cluster not only documents containing ordinary texts, but *any* kind of multidimensional operationalizable information. For the purpose of illustrating both this and the effect of different value assignment to the indexing terms at the same time, we fed the CA with descriptions of 27 flags. This description language consists of a limited vocabulary dealing with the colours, fields, figures, and symmetry of the flags. Figure 7 contains two dendrogram variants. Which one is the “best”?

We claim that the answer to the question just posed above depends on the way the user prefers to perceive the data. A fact which is to be kept *very* much in mind is that there are *so* many options before arriving at the final clustering of whatever items one has. You cannot just “do some cluster analysis” on a material – or if you do, you run the risks of feeding the program with badly cleaned data and running a less appropriate default-parameter-setting for the cluster analysis. It is a little bit like cooking – it takes knowledge both of the ingredients and what you expect at the end. Cluster analysis is, we also claim, like most other less trivial statistical methods; you cannot apply them mechanically on your material and expect their “good reputation”, their “frequent being referred to”, or “guaranteed performance” to make up for dirty data or to having been applied on data of a type they were never meant for. The quality of the treatment of “wild” data from the humanities, such as language, with mathematical instruments will be so much higher if both these scientific disciplines have a feeling for the other.



What's the interest then of creating dendrograms for indexing terms? Maybe the most immediate purpose is that of data exploration, i.e. seeing both what the whole document collection is all about and seeing how the words group, based on how they are used in running text in natural language. Only by looking at Figure 6 we get a rough idea of what is going on in different places in Europe according to this relatively tiny document sample. This study is also valuable for the construction of customized, data-driven thesauri; the lexicographer who does this work sees how the thesaurus should be designed to best capture the reality expressed by a given document collection (presumably a much bigger one than this sample to make such a manual effort worthwhile).

2.7 Cellular projections of the dendrograms and indexing terms

Dendrograms are quite intuitive and reasonably easy to generate and print, even in simple ASCII format. They do however have an upper limit as to how large they may be in order to be printed on an ordinary A4 sheet. The 1,600 document sample referred to here seems to be very close to this limit, printing onto a landscape-oriented paper, even using font size 4 (!). Being able to print on an A3 would probably double the capacity to about 3,000 items, i.e. documents or indexing terms. The CA can however cut up the trees in whatever user-given number of subtrees and print them separately. This subdivision of the tree is done by starting from the root of the tree and then iteratively dissolving the weakest nodes until the desired number of subtrees are left. This method asserts that the most stable subtrees/clusters are kept intact the longest and the dendrogram can be written as a series of trees, one after another. Having this additional possibility, we nonetheless develop a complementary visualizing technique to cope with larger trees (summarily described in [Hagman & al. 99]).

In order to visualize dendrograms in a more compact way we can map it onto a 2-dimensional grid. The procedure is to first cut up the tree into nine subtrees, as described above, and then to distribute these within a 3x3 grid in such a way that more related subtrees come closer to each other. Trying this on a tree containing 260 indexing terms from another data set, we come up with the distribution illustrated by Figure 8:

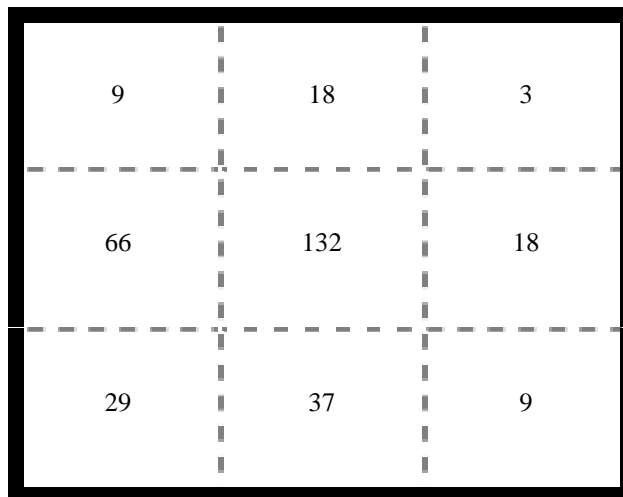


Figure 8 A 3x3 grid where each cell contains one of nine subtrees into which a dendrogram of 260 terms has been cut up. The distribution reflects optimally the intra-subtree relations and the numbers refer to the size of each subtree in terms of words.

In a next step we continue this subdivision and create nine subtrees out of each of these nine we already have. During this subdivision, shown by Figure 9, the “sub-subtrees”, so to speak, are arranged by the same method deciding the arrangement of the subtrees in the original 3x3 grid. We recall that the assignment of each subtree to a cell is done on basis of that subtrees’ relation to the other eight “sister subtrees” in order to make the subtree distribution over the cells optimally reflect their internal similarity. At this phase, however, even some of the surrounding “cousin subtrees” influence the calculation as can be seen in Figure 10 where the subtree in the cell marked with a white hexagon is related to its closest neighbours. The cells are subdivided in the order indicated in Figure 9, i.e. starting with the centre cell and then the method proceeds first +clockwise and, finally, x-clockwise. The reason for starting +clockwise is that we then have one or more whole sides of cousin subtrees already assigned a cell which we can use when determining the best cell assignment for the subtree in question. This cannot be done would we start by going x-clockwise, but the latter x-round can draw all benefits from the result of the former +round.

The final result of this further subdivision is shown by Figure 11. We note that in the cell in the upper right corner the original subtree was not big enough, i.e. did not contain enough many terms/leaves to be subdivided further, so we just left it as it was. In this figure we also indicate the similarity between each neighbouring subtree pair by means of thickness of the line separating the two cells: the thinner, the more similar (i.e. “separated by thinner walls”). Again, the number of words contained in the subtrees/cells is indicated and to enhance the impression grey shades are added.

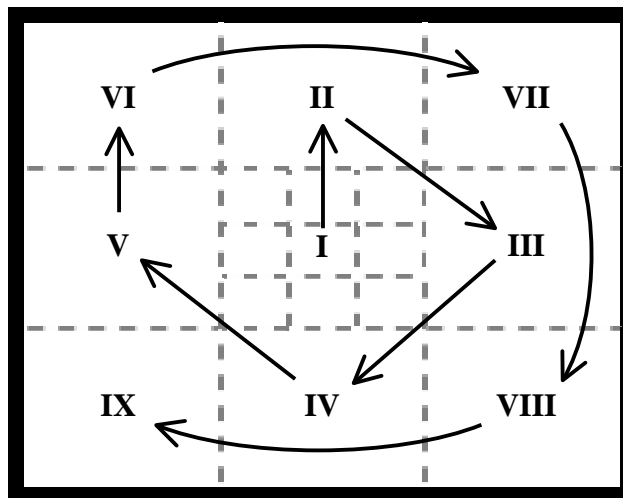


Figure 9 The cells of the 3x3 grid of Figure 8 are collapsed each into another 3x3 grid (where possible), starting with the central cell and proceeding as indicated by numbers and arrows.

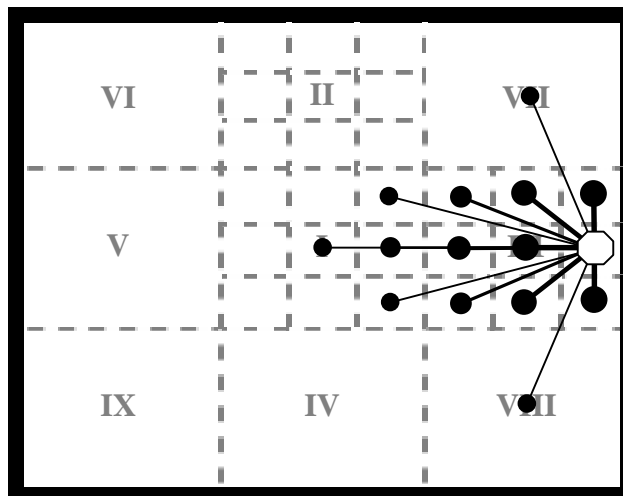


Figure 10 When deciding which cell best suites a given subtree, other 3x3 grid-internat and 3x3 grid-external cells and their already assigned subtrees are related to. Most information is available when carrying out this operation in the order suggested by Figure 9.

1	1	1	2	2	3	3		
1	1	1	2	2	2			
1	1	1	2	2	1			
2	11	4	38	21	13	1	1	1
1	17	8	12	13	7	3	6	1
7	4	12	17	4	7	2	1	2
5	4	3	4	1	4	1	1	1
4	1	3	4	9	6	1	1	1
4	2	1	3	2	2	1	1	1

Figure 11 Eight of the cells of the 3x3 grid of Fig. 8 are now themselves subdivided into a 3x3 grid. The numbers and gray shades indicate the size of the subtrees assigned to the cells. We note that most of the subtrees fall into the left-central area.

Note that at present, the CA only calculates the subdivision and the optimal distribution of the subtrees over the cells in these maps and gives the result in numeric format. It also calculates the thickness of the cell membranes and presents it all as ASCII data, ready to feed into a separate visualization kit.

Figure 12 is a variant of Figure 11 without the number indicating cell size. We notice that the strongest correlating word(group)s occur with a low frequency in this data set as the “walls” between their cells are thin or even non-existent. Looking at this map, a scale drawing of a house comes to one’s mind.

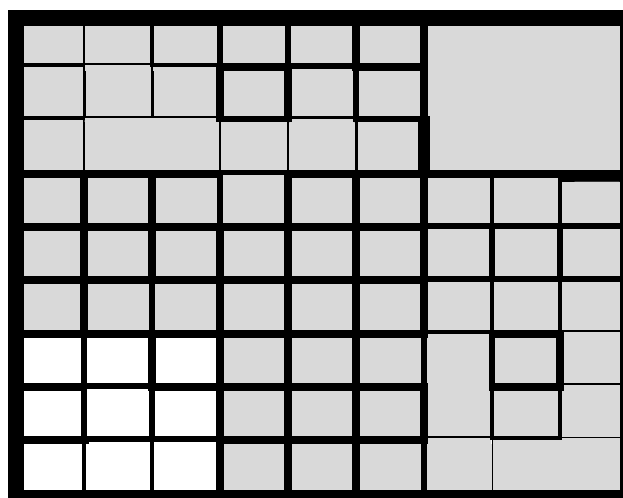


Figure 11 Same as Fig. 9, without density indicators and with all subtrees shaded except the one zoomed into in Figure 11.

Figure 11 is shaded except for the parts occupied by the subtree which Figure 12 zooms into, below, showing the words in its cells. Although the subtrees/cells contain indexing terms in this visualization example, we invite the reader to imagine how this work analogously when visualizing documents.

bovine bse encephalopathy spongiform consumer	antibiotics human addictive feed	limburg protection netherlands
disease orphan medicine product	intend	detect report residue use animal
infect scrapie scientific veterinary	labelling transparency	committee

Figure 11 Zoom in of lower left corner of Figure 10.

Of course, as soon as the document sample size grows beyond almost trivially small sizes, this 9x9 grid must be collapsed further (e.g. into an 81x81 grid, thus approaching the idea of Kohonen maps [Kohonen 95] but retaining the underlying tree structure) and/or the cells made clickable to have their underlying subtree pop up in a session of data exploration or information retrieval. Another path we are likely to explore is to continue letting the CA calculate the dendrogram and optimized arrangement of subtrees and indicators of how these relate to each other, but then export all this to some commercially available visualization tool. In fact, even internally, in our JRC group, we see interesting possibilities of exporting this data as GIS data and in such an environment visualize and explore not geographical maps but document maps and the semantic fields of indexing terms.

3 Current state and usage of the CA

Although the first version of CA has been in service since over eight months, it still lodges in the dockyard between its missions. This is far from unusual in software development where one starts from scratch and builds a system of CA's complexity, designed to meet very special requirements as for data generality and algorithm flexibility. It is a prototype still tricky for other than its constructor to operate. This is because all hitherto available programming efforts have been entirely dedicated to developing and trimming its three most essential parts: (1) the data pre-processing procedures (which, it was discovered, was needed for heavy work as the data was unexpectedly dirty); (2) the highly complex core part, the cluster analysis engine; and (3) the presentation of the results in form of various lists, dendrograms, and cell map coordinates, a part which we also can call 'the data visualization part'. There is as yet no user interface and until the quality and the format of the data we are working with has stabilized, it is not trivial to create a stand-alone executable to be run by an external command with a set of parameters since it is not even clear what variables should be parametrizable. The source code is well modularized although the possibility C offers to write "compact" code has been exploited somewhat to have a better overview of the code itself and to optimize efficiency after compilation. Having a well modularized code is also a must when organizing the procedures and functions into a well structured C code libraries but, again due to the need of modifying many different parts of the program at this stage, up till now it has been convenient to access the whole code (2,200 lines excl. ext. libraries) in one single file.

References

- Barbas, T. and R. Steinberger (1999)
 "New information infrastructures", in *Institute for Systems, Informatics and Safety; Annual report 1998*, EUR 18721, ISBN 92-828-6645-9, European Commission 1999 (pp.25-26).
- Hagman, J., R. Steinberger, D. Perrotta, A. Varfis (1999)
 "Approaches to Document Classification and Visualisation", *16th International Conference on Artificial Intelligence, IJCAI'99*, Stockholm, Sweden, 31/7 - 6/8, 1999.
- Kohonen, T. (1995)
Self-Organising Maps. Springer, Berlin.
- Scott, M. (1999)
WordSmith Tools version 3.0, Oxford University Press, Oxford, UK.
<http://www1.oup.co.uk/elt/catalogu/multimed/4589846/4589846.html>