***Compare sequences***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

<https://www.spsstools.net/en/KO-spssmacros>

All rights reserved

*Compare sequences or documents.* Comparison of sequences of elements with computation of a pairwise similarity measure between the sequences. In comparison, one may take or not take into account chains of elements or their location – i.e., compare as sequences of consecutives or simply as documents. You may choose which way to establish the similarity by: maximal matching, simple aligning, maximal common chain, etc.

*Read “*[*About SPSS macros*](https://www.spsstools.net/en/KO-aboutmacros)*” what are they and how to run them.*

*The “Protected directory” error.* Some of the macros described in the current document write temporary files to hard disc. If you don't have full Administrator rights of your computer, it may cause error saying, among things: *“SPSS Statistics cannot access a file... specifies a protected directory...”*, meaning that the default directory the macro wants to use is protected on your PC. To solve the problem, in Syntax window issue command: CD 'myfolder'., where 'myfolder' is the path/name of some folder where you are allowed to save files to.

# MACRO !KO\_SEQSIM: COMPARING SEQUENCES (OR DOCUMENTS), AND THEIR SIMILARITY MEASURE

Version 3, Jul 2023 (Version 1, Jun 2014). Tested on SPSS Statistics 22, 26, 28.

!KO\_seqsim docvar= *seq* /\*Short string identifier variable for sequences

/wordvar= *word* /\*Variable of "words", string or numeric

/compare= FIRST /\*Task: compare sequences all in pairs and compute similarities

/\*(ALLPAIRS, default), compare the first with every other and compute similarities

/\*(FIRST), or compare the first with every other and extract the longest

/\*common chain (FIRSTEXT)

/trv= /\*Optional: set term relevances: file/dataset or AUTO

/trsm= /\*Optional: set term relevances and similarities: file

/trotr= /\*Relevances for "other" terms (default =1)

/chmaxw= *4* /\*Maximal weight to words depending on chain length: number

/\*or NOLIM (no limitation, the observed length);

/\*may use keyword SQUARE after

/chminw= *1* /\*Minimal weight to words depending on chain length: number;

/\*may use keyword INI after

/chpass= /\*If TRV/TRSM specified and CHMAXW<>1: one/two pass scores,

/\*for dichotomizing before chain highlight

/chbward= /\*Consider chains with backward order: NO (default) or YES

/diag= /\*Optional: check diagonality of co-occurrence matrix: MAIN, MIDDLE, or BAND

/diagls= /\*Optional, for DIAG: parameters location and shape in "penalty function"

/rescrnd= /\*If DIAG or TRV/TRSM specified: rescale/round co-occurrence weights,

/\*for speed of the matching

/method= GREEDY PRINT /\*Similarity reckoning method: greedy matching (GREEDY, default,

/\*may use keywords REORD, PRINT after GREEDY); Hungarian matching (HUNGAR);

/\*maximal trace (MXTRACE); maximal chain (MXCHW); maximal weight (MXW);

/\*mean positive weight (MEANPW); sum of weights (SUM)

/dlen= /\*Normalizing length in the formula denominator: MIN (default), MAX, MEAN, GMEAN,

/\*HMEAN, NONE (raw similarity)

/divisor= /\*Normalizing deflator of co-occurrence weights: DIVISOR1, DIVISOR2,

/\*DIVISOR3 (default), DIVISOR4

/wmax= /\*Optional, Wmax for deflator: positive integer

/savem1= /\*Optional, if two sequences: save initial co-occurrence

/\*matrix (path/file or declared dataset for saving)

/savem2= /\*Optional, if two sequences: save co-occurrence matrix

/\*before reckoning similarity (path/file or declared dataset for saving)

/savemxaw= *'d:\exercise\mxw2.sav'* /\*Optional: save maximal weight in each

/\*comparison, obtained at

/\*chain highlighting (path/file or declared dataset for saving)

/print= MEDIUM /\*Optional: dynamic printout (SHORT, MEDIUM, or LONG)

/dataset= *data* /\*If COMPARE=FIRSTEXT or TRSM specified: input dataset name or RENAME

/newdata= /\*Optional: name for the output dataset.

Minimal specification DOCVAR, WORDVAR, METHOD.

What’s new Sep 2023:

Chains highlight with long sequences now runs faster. Chains highlight speed now does almost not depend on CHMAXW parameter.

The macro compares, in pairs, sequences of some elements. Sequences are similar if (first) the compound of elements, *content*, is alike in them; but they are especially similar if (second) herewith the sequential *order* of *contiguous* elements is alike in them. Comparison just by compound is the “documents comparison”, and taking account of the order is the “sequences comparison”. The macro lets you choose, or tune, to what degree it should consider the similarity of consecutive order above the similarity just of content. You may also detect and save the most long chain (series of consecutive elements) with the same order of elements in the two sequences compared.

The input data to the macro are two variables (see particulars in the description of subcommands DOCVAR and WORDVAR). One is the document (sequence) identifier and the other is the elements per se (we’ll call them “words”, for convenience); each dataset case is a “word”. The order of words in each document – i.e., the case order in the dataset – is important: it must be the one existing in the sequences you study. But if you are going to compare documents purely by content, i.e., without paying attention to adjacent words order, the order is unimportant.

*Attitude to gaps (“punctuations”) at comparing sequences*. You are permitted to have gaps between elements in the input data (for example, CD∙ABA∙B) – see WORDVAR subcommand. However, the macro takes positions of gaps for fixed at input. The consequence of this will be that CD∙ABA∙B and CD∙AB∙AB – are not identical sequences for the macro. This macro does not shift words apart or towards in order to “establish” the maximal congruence of the sequences. And the present macro does not do sequence comparison known as *optimal matching* – which utilizes edit operations (insert, delete, substitute) to obtain the best “global” alignment.

*If you are studying sentences of a natural language*. !KO\_SEQSIM does not engage in linguistic stemmatization / lemmatization, leaving this preliminary work on the user. Also, upper and lower letters in the data are different letters for the macro, it does not ignore case.

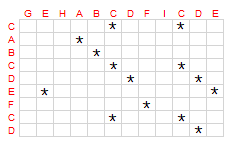
The macro outputs its main result – similarity measure or chains – as a new unnamed dataset (there is an option to give it a name). By default, square matrix of similarities between all sequences is returned. In the input dataset, the macro creates variable *WORD\_.#$*, and sometimes also *NSEQW\_.#$* and *RELEV\_.#$*. Subsequent runs of the macro delete and re-create these variables.

**Brief overview of the macro**

Co-occurrence matrix

There exist many methods and algorithms to compare documents or to compare sequences. They have different ideologies and applications. The present macro realizes some of approaches based on processing of *co-occurrence matrix*. The co-occurrence matrix between elements of two documents/sequencies, for example, CABCDEFCD and GEHABCDFICDE, looks like

**Fig. 1**

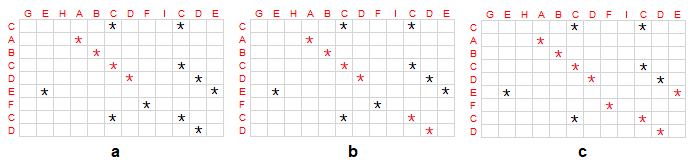


It is in fact the adjacency matrix corresponding to a bipartite graph. Weights of co-occurrences, here units (shown as asterisks), may as well be other positive numbers – that depends on how you specify input parameters for the macro. Section “Algorithm” describes in detail all the stages of the analysis. The macro analyzes the matrix as is, fixed; it does not move positions of co-incidences to and fro and does not apply edit operations (insertion, deletion, etc.) to sequences as tools to “find” the maximal congruence.

Various approaches to determine the similarity

What is the similarity between these two documents (or sequences)? Many ways to calculate, reckon it, exist. Here is what the macro uses. One approach (**Fig. 2 a**) is to find out the length of the longest chain of contiguous co-occurrences. It is the method of *maximal common chain*. Evidently, this approach regards comparison of documents as a comparison of sequences of adjacent “words”. Another approach (**Fig. 2 b**) is to find out the greatest diagonal (slant) sum of co-occurrences. It is the method of *maximal aligning* (in its simple, basic form). The approach regards comparison of documents as a comparison of sequences of “words”, but not necessarily adjacent. A third approach (**Fig. 2 c**) is to take co-occurrences of the matrix into account maximally, still pairing by the principle: one row of the matrix can pair with no more than one column, and vice versa. It is the method of *maximal matching*[[1]](#footnote-1). This approach treats comparison of documents wider – as a comparison of texts (compound of “words”), where the sequence is not necessarily important. But you can strengthen its sensitivity to sequences of adjacent words, if you request to value (weight) longer chains, coinciding in the two documents, more. Of the differentiation of weights of co-occurrences – see further.

**Fig. 2**



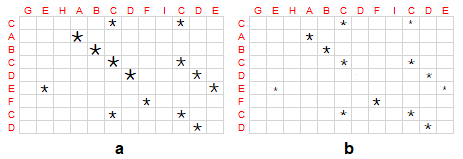
Besides the three mentioned approaches to determine the size of similarity (which shared feature is that they three “pair” words of the documents being compared by the one-to-one principle), the macro offers some *other*, simpler approaches to reckon the similarity. Say, you may just count/sum up all co-occurrences in the matrix – and that will be similarity magnitude of the documents. Or you may have out one, maximal weight in the matrix. Or average weight. (Two last modes make sense if co-occurrences vary in weight.)

Approaches on which !KO\_SEQSIM is based are more frequently employed in social studies than, put it, in molecular biology. You may well use the macro in text analysis (after appropriate preparation of the data).

Different weights to co-occurrences

You may give different weights to co-occurrences in the co-occurrence matrix. First (**Fig. 3 a**), may “award” (increase weight) chains of contiguous words, coinciding in the documents, for their length (you may also “punish” chains insufficiently long for you). If you are doing this – that means you are treating, to this or that extent, document comparison as comparison of *sequences* *of adjacent* words. Second (**Fig. 3 b**), you may “punish” (decrease weight) of co-occurrences in those rows and columns of the matrix where co-occurrences are present far away from the diagonal position. If you are doing this – that means, for you, similar documents are almost copies of each other: you demand not only coincidence by words and/or word chains, but also by *disposition* of the coincident parts. Third, you may put different weights to co-occurrences from the beginning – dependent on the *relevance* of specific words for you in the task of comparison (you may also set a nonzero weight to co-occurrences of some nonidentical words, if they are “synonyms” to you).

**Fig. 3**



EXAMPLE 1.

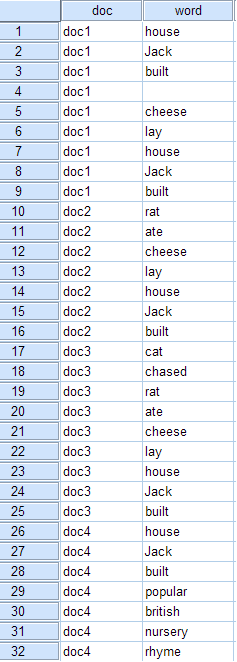
doc1 = “This is the **house** that **Jack built**. This is the **cheese** that **lay** in the **house** that **Jack built**.”

doc2 = “This is the **rat** that **ate** the **cheese** that **lay** in the **house** that **Jack** **built**.”

doc3 = “This is the **cat** that **chased** the **rat** that **ate** the **cheese** that **lay** in the **house** that **Jack built**.”

doc4 = “This Is the **House** That **Jack Built** is a **popular British nursery rhyme**”.

Let’s take only the bolded words and create the dataset ready for work:



The gap between **built** and **cheese** in doc1 is added for the reason that here is a stop between sentences, we don’t want ‘**built cheese**’ be treated as a chain of consecutive words.

data list free /word (a10).

begin data

house Jack built ' ' cheese lay house Jack built

rat ate cheese lay house Jack built

cat chased rat ate cheese lay house Jack built

house Jack built popular british nursery rhyme

end data.

string doc (a8).

do if $casenum<10.

compute doc= 'doc1'.

else if $casenum<17.

compute doc= 'doc2'.

else if $casenum<26.

compute doc= 'doc3'.

else.

compute doc= 'doc4'.

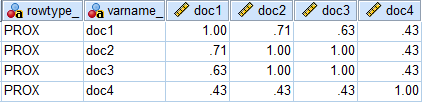
end if.

execute.

dataset name data.

* Data entered.

!KO\_seqsim docvar= doc /wordvar= word /method= GREEDY.



* The macro output square matrix of similarity coefficients between the 4 sequences pairwise. By default, parameters CHMAXW=1, CHMINW=1, what means that only word composition (content) will be considered as the base of likeness, and not the order of consecutive words (not chains of words).
* The method of accrual the similarity selected – greedy matching.
* DLEN=MIN, by default: the length of the shorter of the two being compared documents will be the normalizing denominator in the formula of similarity coefficient.
* Variable *WORD\_.#$*, created by the macro in the input dataset – is *WORD* recoded by AUTORECODE command. A random negative number in the variable corresponds to a gap between words.

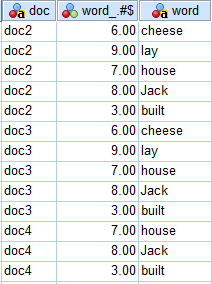
!KO\_seqsim docvar= doc /wordvar= word /chmaxw= 3 /method= HUNGAR /dlen= NONE.

* In this run, CHMAXW=3 renders more weight to words going in the same adjacent order in the documents, specifically: a chain of concurrence of length 3 words or more is valued by weight 3 (for example, ‘**ate cheese lay house Jack built**’). A chain of concurrence of length 2 words is valued by weight 2. Concurrence by an isolated word is valued by weight 1 (in the previous run, there by weight 1 were valued words irrepective of whether they are in chains or isolated).
* Hungarian matching algorithm is used in place of the greedy method.
* Normalizing is set aside (DLEN=NONE): similarity in the raw form, only the numerator of the formula, will be returned. Conventional 999 will be on the matrix diagonal.

!KO\_seqsim docvar= doc /wordvar= word /chmaxw= NOLIM /method= MXCHW /dlen= NONE /print= LONG.

* In this run, CHMAXW=NOLIM awards increased weight to words going in the same contiguous order in the documents, specifically: a chain of concurrence of length *l* is valued by weight *l*. PRINT=LONG displays co-occurrence matrices to the user. Chain highlight is visible in them.
* The method to reckon similarity – maximal common chain. Similarity in a pair of documents will equal the length of their longest common chain, squared (under no normalization, DLEN=NONE).

!KO\_seqsim docvar= doc /wordvar= word /compare= FIRSTEXT /dataset= data.



* In this run, the macro returns not the similarity measure (a number) but the longest continuous subsequence (chain of contiguous words), concurring in documents doc1 and doc2, in documents doc1 and doc3, in documents doc1 and doc4.

**Terminology**

A document for us is a concrete sequence of some elements, *“words”*. We’ll further write without quotes: word. Thus, document ‘A B C B D’ consists of 5 words. We understand *document* and *sequence* as synonyms, unless where specially differentiated. *Term* is a distinct word, a variety kind of words. In sequence ‘A B C B D’ term ‘A’ has one *occurrence* (entry), term ‘B’ – two occurrences, and so on. *Co-occurrence* (or coincidence) – is every instance when a term (or word) of one document meets with itself in the other document. For example, in the pair of documents ‘A B C B D’ and ‘B F B B A’, term A co-occurs 1×1=1 time, but term ‘B’ co-occurs 2×3=6 times. Co-occurrence of a term with another term we would also call co-occurrence[[2]](#footnote-2). *Continuous subsequence* (aka *chain*) – is several contiguous words. In ‘A B C B D’, different subsequences can be extracted, for example, ‘A B C’, ‘C B’, ‘B C B D’ etc. (Synonyms of this concept, encountered in literature: substring, series, run.) *Continuous subsequence* *(chain) of concurrence*, or common chain – is a chain met in one as well as in the other document. So, in sequences ‘A B C B D’ and ‘B F B B A’ there are no concurrent chains (of length more than one word).

**Algorithm**

There exist very different approaches to establish likeness between two documents/sequences. Macro !KO\_SEQSIM offers to select from several approaches. Among them, the following three are interesting:

1. *Maximal matching* approach: similarity between two documents is the magnitude of one-to-one matching elicited from the co-occurrence matrix between these documents. Establishing of the matching consists of counting co-occurrences, whereby one occurrence can give no more than one co-occurrence into the matching (i.e., matching is one-to-one). The goal of maximal matching is to pair between rows and columns in the matrix as much as possible; else speaking – to manage to include as many co-occurrences as we can, into the similarity. Optionally one may recognize (reward) the existence of identical chains, same series of adjacent words, in the documents. In that case, documents similarity that is accrued in the act of matching will represent similarity not only by word content, but also by order of adjacent words: the similarity tends to be the higher the more there are found concurrent chains between the documents and the longer those chains are. Therefore, matching approach can consider or ignore sequence of words in documents.
2. *Maximal aligning* approach: similarity between two documents is the greatest diagonal (slant) sum in the co-occurrence matrix between these documents. This method regards a document to be a sequence of words, albeit not necessarily adjacent. Its goal is different from that of maximal matching, and comes down to finding such a specific, particular adhering between the sequences, wherein the maximal quantity of co-occurrences is discovered. As in the previous case, optionally one may reward the existence of identical subsequences of adjacent words in the documents.
3. *Maximal common chain* approach: similarity between two documents is the longest chain of concurrence (or the weightiest of long chains of concurrence) between these documents. This method regards a document to be a sequence of adjacent words. Unlike the previous approach, diagonal sum confined to only one, continuous subsequence, plays a role here.

These three approaches are the core of the macro’s algorithm. They act on stage 3 “Reckoning raw similarity”. All stages of the algorithm are described next.

The job, the algorithm of comparison of two documents, moves in 4 stages (see also **Fig. 16**):

1. Creation of initial co-occurrence matrix. Optional application of relevances/similitudes.
2. Optional altering of weights of co-occurrences (any of the 3 substages can be omitted):
   1. Weighting by contiguous words’ chain length: a co-occurrence that is a link in a concurrent continuous subsequence is valued (rewarded) higher than an isolated co-occurrence. Use this substage when you regard the documents being compared as sequences of adjacent words. The main subcommand guiding the substage is CHMAXW.
   2. Checking diagonality: a co-occurrence is penalized by weight decrease if co-occurrences in its row and its column deviate from the diagonal position. Use this substage if, in the documents being compared, the general disposition of parts coinciding in them is important to you. The subcommand in charge of this substage is DIAG.
   3. Rounding weights with the goal of discretizing them (optional and of technical character).
3. Reckoning raw similarity (the numerator). Selection of co-occurrences for summing up their weights into the raw similarity – is done via the method you select.
4. Normalizing (division by the denominator) into the similarity coefficient (optional).

The stages are considered below in detail.

1. **Creation of initial co-occurrence matrix**

Let sequence D1 be CDABAB, sequence D2 be EABACCCBB. A character corresponds to a “word” here. (A word can be any value, numeric or string; the macro always recodes values at input in numbers 1, 2, ... and works internally with them already.) Create binary *co-occurrence matrix* **M**, where 1 marks each co-occurrence: a word of one document met with the same word of the second document. Co-occurrence matrix is a binary adjacency matrix of a bipartite graph.

**Fig. 4**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 1 | 1 | 1 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 1 |  |  |  |  | 1 | 1 |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 1 |  |  |  |  | 1 | 1 |

The matrix is binary – empty cells are shown on the figure instead of zeros.

*Co-occurrence matrix in situation of TRV or TRSM subcommands*. Matrix **M** may be created not binary, but rather consisting of values of range [0,1], if s/c TRV or TRSM is specified. Under TRV condition, co-occurrences of same words can be of weight from 0 to 1; these values are called term relevances. Under TRSM condition, the same situation with relevances takes place, plus to this, co-occurrences of not same words can be nonzero, they can be of weight from 0 to 1; these values are called term similitudes (similarities). Relevances and similitudes for terms are provided by the user, as a reference, but there is an option to compute relevances automatically. Here is an example how **M** might be looking under TRSM specified:

**Fig. 5**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C | 0.1 |  |  |  | 0.6 | 0.6 | 0.6 |  |  |
| D |  |  |  |  |  |  |  | 0.2 | 0.2 |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 0.8 |  |  |  |  | 0.8 | 0.8 |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 0.8 |  |  |  |  | 0.8 | 0.8 |

Empty cells are shown on the figure instead of zeros. Note that some co-occurrences of terms not with themselves – similitudes – are nonzero.

(Relevance and similitude are concepts in the same line, juxtaposed: relevance is the importance of co-occurrence of the term with own self, and similitude is the importance of co-occurrence of two different terms. In either case the importance of co-occurrence plays “for” likeness of the being compared documents, if it is close to 1, and plays “against” their likeness, if it is close to 0. By default, i.e. without s/c TRV/TRSM, relevance of every term is assumed 1, and all similitudes between terms are assumed 0.)

See s/c TROTR explaining how giving a term relevance 0 is like and how is different from physical deletion of the term from data.

1. **Altering of weights of co-occurrences**

This stage or any of its substages can be skipped by user decision.

1. Weighting by length of concurrence chain

If CHMAXW=1 (which is the default) this substage is skipped and **M** transits to substage 2b. With CHMAXW=1, concurrence chains of any length have weight 1, the same unit weight as individual elements of the co-occurrence matrix. That is, concurrence chains between D1 and D2 are not valued higher than separate co-occurrences. Consequence of it: there will take place comparison of mere word compound, document content, and not adjacent word order in the documents.

If CHMAXW is greater than 1 or CHMAXW=NOLIM, then elements forming concurrence chains are given *weight* greater than 1. That means the order of adjacent words will be taken into account. With CHMAXW=*n*, elements of concurrence chain of length ≥*n* will be highlighted (rewarded, weighted) by integer weight *n*, elements of concurrence chain of length *n*-1 will be highlighted by weight *n*-1, elements of concurrence chain of length *n*-2 will be highlighted by weight *n*-2, and so on. Value *n* is the ultimate weight. But if CHMAXW=NOLIM, no explicit limit is laid by the user, and the elements of a concurrence chain will be highlighted always by the weight equal to the chain’s length.

Under CHMAXW=2, our matrix **M** (**Fig. 4**) becomes:

**Fig. 6**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 1 | 1 | 1 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 2 |  | 1 |  |  |  |  |  |
| B |  |  | 2 |  |  |  |  | 1 | 1 |
| A |  | 2 |  | 2 |  |  |  |  |  |
| B |  |  | 2 |  |  |  |  | 1 | 1 |

and under CHMAXW=3 or NOLIM our matrix looks this:

**Fig. 7**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 1 | 1 | 1 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 3 |  | 1 |  |  |  |  |  |
| B |  |  | 3 |  |  |  |  | 1 | 1 |
| A |  | 2 |  | 3 |  |  |  |  |  |
| B |  |  | 2 |  |  |  |  | 1 | 1 |

We observe presence of common chains ABA and AB between D1 and D2, and in case CHMAXW=3 or NOLIM the chain of 3 words will be praised higher than the chain of 2 words.

If there is specified CHMINW=*m*, concurrence chains of length <*m* get permanently erased from **M**: co-occurrences that form them won’t be considered further in the algorithm. Only co-occurrences of length ≥*m* can be the base of similarity. Thus, if CHMAXW=3 and CHMINW=2, our **M** presents as such:

**Fig. 8**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 3 |  |  |  |  |  |  |  |
| B |  |  | 3 |  |  |  |  |  |  |
| A |  | 2 |  | 3 |  |  |  |  |  |
| B |  |  | 2 |  |  |  |  |  |  |

Co-occurrences forming chains of length less than 2, i.e., not forming chains in this case, have been erased.

Using of keyword INI in CHMINW=*m* does not erase but keeps weight 1 for co-occurrences forming concurrence chains of length <*m*. In this case, and in our specific example, **M** will come out like it is on **Fig. 7**, not on **Fig. 8.**

*Note 1*. With COMPARE=FIRSTEXT, the macro performs substage 2a in CHMAXW=NOLIM regime, with the aim to extract the longest concurrence chain, and then exits the algorithm.

*Note 2*. With CHBWARD=YES, highlighted will be not only chains going top-left to bottom-right, but also going top-right to bottom-left. Then a matrix element receives the weight larger of these two results in its cell.

*Resume*. Weighting by the chain length is the multiplication of initial weight *w* of the co-occurrence (i.e. 1) by the factor (integer, ≥1) directly representing the length of the concurrence chain to which that co-occurrence belongs. The co-occurrence starts to weight (*w*) as a representative of a chain of a certain length. Weighting (= highlighting, rewarding) of chains transfers document comparison on the rails of sequence comparison, namely, taking into account the order of contiguous words.

*Weighting by chain length in situation of TRV or TRSM subcommands*. The described substage 2a, the rewarding of concurrence chains, needs a binary **M** entering it (1=co-occurrence, 0=not-co-occurrence). When **M** matrix was created under dictation of TRV/TRSM subcommands, then most likely it is nonbinary, consisting of fractional values in [0,1] bounds. Then it is necessary first to dichotomize it. Values ≥ pass level (it is indicated by the user in CHPASS subcommand) are recoded into 1, and the rest – into 0. After that, rewarding of chains is being done in the manner described above. When the highlighting is ready, so that **M** contains integer weights, it will be multiplied, elementwise, by the initial **M** with fractional weights. Therefore, after exiting substage 2a weights will become fractional again.

Let us take the matrix on **Fig. 5**. With CHMAXW=3 and the passing score CHPASS= 0.7 0.2 (the 1st value – for relevances, the 2nd – for similitudes) the matrix will appear as shown here, after chain highlighting and final multiplication by initial weights:

**Fig. 9**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  | 0.2 | 0.2 |
| A |  | 3 |  | 1 |  |  |  |  |  |
| B |  |  | 2.4 |  |  |  |  | 0.8 | 0.8 |
| A |  | 2 |  | 3 |  |  |  |  |  |
| B |  |  | 1.6 |  |  |  |  | 0.8 | 0.8 |

If you need to know the maximal weight in **M** before exiting the substage (and prior the multiplication of that **M** by the initial weights in case of TRV/TRSM specified), address to SAVEMXAW s/c.

*Comment on zero weight*. Co-occurrences (matrix **M** positive elements) admitted to the highlighting of chains and retained a nonzero weight there, can never get zeroed later. In other words, if a chain is recognized (rewarded), it is not fractioned further. This arises from two circumstances: (i) co-occurrences with relevance/similitude 0 are not admitted to the highlight (CHPASS cannot be 0), consequently, the “elementwise multiplication by the initial **M** with fractional weights”, introduced two paragraphs above, cannot turn a nonzero weight rewarded in the highlighting “back” into a zero one; (ii) further substages 2b (DIAG subcommand) and 2c (RESCRND subcommand) do not zero co-occurrence weights. A zero weight in **M** at the entry to stage 3 (reckoning similarity) – is either (i) a zero weight from the start, in the initial co-occurrence matrix, or (ii) a co-occurrence having lost entirely its weight in the substage 2a (chain highlight) due to s/c CHPASS (co-occurrence was not admitted to highlighting) or due to s/c CHMINW (co-occurrence was killed as a result of highlighting).

1. Check of diagonality

With subcommand DIAG not specified (what is the default), this substage is skipped and **M** passes over to substage 2c.

DIAG reduces similarity of the documents if they are not copies of each other. A user imposes this reduction if, for them, similarity between documents is large only provided that their words or word subsequences identical between them occupy similar positions inside the documents. It is the situation, where co-occurrences or chains of co-occurrences in **M** matrix gravitate towards diagonal location, while locations away from the diagonal are predominantly empty.

For example, the user may reason that, albeit the number of co-occurrences on the left and the right sides of the following figure is equal (five each), similarity on the left is actually higher than on the right, because on the left panel individual words or chains formed of them are found in about the same sites in both documents (so the structure is close to be diagonal), while on the right panel – they are found in the different sites:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | C | D |  |  | C | D | A | B | C |
| A | 1 |  |  |  |  |  | A |  |  | 1 |  |  |
| B |  | 1 |  |  |  | vs | B |  |  |  | 1 |  |
| C |  |  | 1 | 1 |  |  | C | 1 |  |  |  | 1 |
| D |  |  |  |  | 1 |  | D |  | 1 |  |  |  |

Or, for instance, the user may consider similarity on the next figure to be higher on the left, than the right, for the reason that words on the left don’t repeat and therefore occupy unique positions, whereas on the right there is deficient word diversity, hence – lack of unique positions:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |  |  | A | A | A | A | E |
| A | 1 |  |  |  |  |  | A | 1 | 1 | 1 | 1 |  |
| B |  | 1 |  |  |  | vs | A | 1 | 1 | 1 | 1 |  |
| C |  |  | 1 |  |  |  | A | 1 | 1 | 1 | 1 |  |
| D |  |  |  | 1 |  |  | A | 1 | 1 | 1 | 1 |  |

In both these examples, check of diagonality will “punish” the variants shown on the right: the similarity will be lower for them in the end. Variants on the left are closer to that state of affairs when the documents are copies of each other. Without check of diagonality, the macro will consider so that on the two right side pictures the documents have the same similarity magnitude as there is on the left side pictures (suppose, when the similarity is accrued by maximal matching way).

Diagonality check is done as follows. The matrix of deviations from the diagonal, offset matrix **O**, sized as **M**, is created. Here is an offset matrix 5×5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 1 | 0 | 1 | 2 |
| 3 | 2 | 1 | 0 | 1 |
| 4 | 3 | 2 | 1 | 0 |

If **M** matrix is nonsquare, offset matrix will be the one the user prefers in DIAG subcommand.

DIAG=MAIN. The main (left) diagonal is taken for the base:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 2 | 3 | 4 |
| 2 | 1 | 0 | 1 | 2 | 3 |
| 3 | 2 | 1 | 0 | 1 | 2 |

DIAG=BAND. All diagonals are taken for the base, the band:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 2 | 3 |
| 1 | 0 | 0 | 0 | 1 | 2 |
| 2 | 1 | 0 | 0 | 0 | 1 |
| 3 | 2 | 1 | 0 | 0 | 0 |

DIAG=MIDDLE. Only the middle diagonal of the band is taken for the base:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 2 | 3 | 4 |
| 2 | 1 | 0 | 1 | 2 | 3 |
| 3 | 2 | 1 | 0 | 1 | 2 |
| 4 | 3 | 2 | 1 | 0 | 1 |

If the band consists of even number of diagonals, there are two middle diagonals.

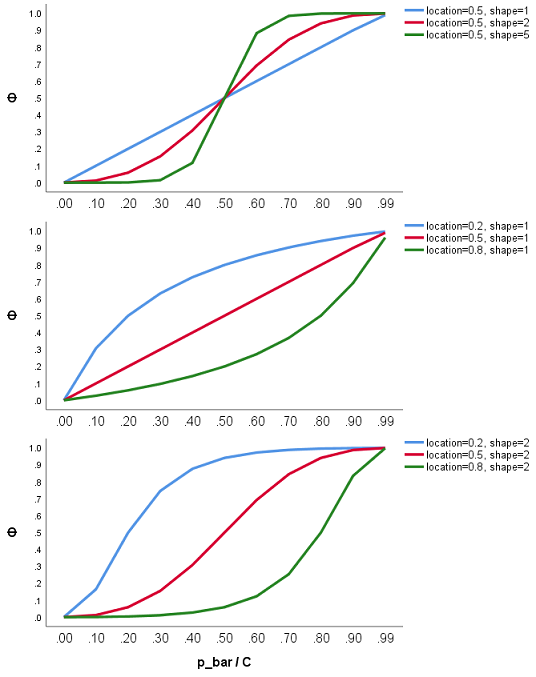
Let **P** be **O** matrix in which those elements are zeroed that are zero (empty) in the **M** matrix entering the 2b substage. Then

where is the average of nonzero elements in row *r* of matrix **P** (0, if no such elements); constant *C* is the number of columns in **P**; is the average of nonzero elements in column *c* of matrix **P** (0, if no such elements); constant *R* is the number of rows in **P**.

“Penalty” function Φ by default – it is the identity function (i.e., its argument itself), which corresponds to the straight, linear dependence of “penaly” on the size of deviation from the diagonal. But one may demand a curvilinear dependence. In general case,

and Φ(*x*) = 0 if *x* = 0

where *x* is from [0,1) domain, *l* is the curve location parameter: number in the range (0,1), and *s* is the parameter of shape, or steepness, of the curve: number ≥ 1. The parameters are specified by s/c DIAGLS. With *l* = 0.5 and *s* = 1 we have the straight line, the identity function, as by default. On the picture below, curves of Φ(*x*) are shown, under various values of parameters of location *l* and of shape *s*. The higher is the curve, the greater is the penalty for non-diagonality. Strongly sigmoid curve implies contrasting attitute: tolerance to small deviations and intolerance to large ones. Function Φ is a specific and flexible variant of the logistic function[[3]](#footnote-3).



The weight of a co-occurrence, , as it was before the 2b substage, gets updated by multiplying it by the factor . And that is the check of diagonality, the correction by non-diagonality.

The meaning of this correction is the penalty to co-occurrences, depending of non-concentration of co-occurrences in the given row and the given column. All co-occurrences of row *r* must receive the same penalty, the stronger one the larger is the mean deviation of co-occurrences in that row from the diagonal position in it. And likewise, all co-occurrences of column *c* must receive the same penalty, the stronger one the larger is the mean deviation of co-occurrences in that column from the diagonal position in it. In the result, co-occurrence in cell *r*,*c* receives the hybrid penalty, the geometric mean of the two penalties – if there exists the diagonal position both in *r* and in *c*. But if (due to nonsquare matrix) there exists only one diagonal position in front of cell *r*,*c*, then only one, correspondent, penalty is imposed. Pay attention, that a penalty to a co-occurrence does not depend on deviation, from the diagonal, of this individual co-occurrence, rather, it depends of deviations, from the diagonal, of all the co-occurrences in its row and/or its column: the penalty is a function of the row, function of the column, it is a collective penalty.

If a co-occurrence lies on the diagonal position and there are no co-occurrences in its row and its column lying off the diagonal position, then and only then ; else will be below 1 and, consequently, will be punished by lowering.

To illustrate, let’s show matrix **M** taken from **Fig. 6**. Here is how it will look after reduction of its weights, i.e., multiplication by :

**Fig. 10**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 0.38 | 0.27 | 0.44 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 1.54 |  | 0.86 |  |  |  |  |  |
| B |  |  | 1.30 |  |  |  |  | 0.63 | 0.63 |
| A |  | 1.44 |  | 1.61 |  |  |  |  |  |
| B |  |  | 1.37 |  |  |  |  | 0.70 | 0.70 |

Let’s trace, for instance, co-occurrence B found in cell (4,3). Its weight was (**Fig. 6)** = 2. Let DIAG=MAIN, so **O** matrix is:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 |

And hence **P** matrix is (empty cells are 0):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 4 | 5 | 6 |  |  |
|  |  |  |  |  |  |  |  |  |
|  | 1 |  | 1 |  |  |  |  |  |
|  |  | 1 |  |  |  |  | 4 | 5 |
|  | 3 |  | 1 |  |  |  |  |  |
|  |  | 3 |  |  |  |  | 2 | 3 |

In its row 4 there are three positive elements, and their average divided by the number of columns (9) = 0.37. In column 3 there are two positive elements, and their average divided by the number of rows (6) = 0.333. Therefore, the current weight (2) is being multiplied by and turns into 1.296 ≈ 1.30.

*Resume*. Check of diagonality is multiplication of the current co-occurrence weight by the coefficient (higher than 0 and up to 1 inclusive) characterizing its row its that column in the co-occurrence matrix; the coefficient captures concentration of co-occurrences near the diagonal/s of the matrix. Unlike chain rewarding (substage 2a), diagonality check concerns not a micro-structure of a sequence (in a form of contiguous words), but a macro-structure of a sequence.

1. Rounding co-occurrence weight

With subcommand RESCRND not specified (what is the default), this substage is skipped and **M** passes over to stage (3). The substage is redundant and is always skipped when weights are integer, i.e., when there were no subcommands TRV/TRSM and no substage 2b.

Fractional values of **M**, especially after 2b stage, may tend towards continuity. In this circumstance, speed performance of maximal matching (stage 3, METHOD=GREEDY/HUNGAR) is worse than in the circumstance of modest diversity of discrete weights. RESCRND discretizes weights for the sake speed of the matching, though it leads to some loss of precision of the similarity being computed. Let *dec* be number 1, 2, or 3, specified in RESCRND. Let *resc*=max(**M**). The macro divides values in **M** by *resc*, what is the rescaling of range 0-*resc* into range 0-1. Then rounds the values to *dec* decimal digits, and then multiplies by *resc*. If a nonzero value becomes 0 as a result of the rounding, it is replaced for 10-9.

*Note*. Substage 2c is unnecessary for methods not matching ones, because the speed does almost not depend then on discreteness of weights. However, you may utilize the substage in this case either.

*Resume*. This is discretization of weights by a linear rescale-round method.

1. **Reckoning raw similarity**

It is the key stage: reckoning of (raw) similarity magnitude *S* between documents D1, D2. It is directed by METHOD subcommand. The user is suggested several methods to choose from, they express different approaches to the concept “similar sequences/documents”. Say, the idea of matching is to account *as many as possible* of word coincidence between the being compared sequences, without duplication (one word can pair just once). The idea of aligning is to find such an adherence of two subsequences, one along the other, in which there are maximally many word coincidence found *against each other*.

*Maximal matching approach*

Utilized is a (some) algorithm of computing the maximal matching (pairing) between the sets of a bipartite graph. The idea is to pair rows with columns in **M** in a way to hopefully *maximize* the sum of elements (co-occurrence weights, *w*) on the crossing of the paired rows/columns. This sum is the magnitude of similarity between the two sequences.

Essential is the one-to-one nature of matching: a row can pair no more than with one column, a column can pair no more than with one row. To put differently, one occurrence of a term in one document can pair (i.e., to vote for the similarity) just with one occurrence of this[[4]](#footnote-4) term in the other document. If one document is, say, “A B A C”, and the other is “A A D A D A”, the magnitude of similarity equals 2: each of two A of the first document has found for itself exactly one A in the vis-à-vis document, to pair with. In general, each element of one sequence is given a chance to find and pair with single element of the second sequence; and if that latter is already paired, the chance vanishes. Consider examples.

With CHMAXW=1 **(Fig. 4**), the matching will turn, most likely, this:

**Fig. 11**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 1 | 1 | 1 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 1 |  |  |  |  | 1 | 1 |
| A |  | 1 |  | 1 |  |  |  |  |  |
| B |  |  | 1 |  |  |  |  | 1 | 1 |

with the similarity magnitude *S*(D1, D2) = 1+1+1+1+1 = 5.

With CHMAXW=3 (**Fig. 7**), the matching will turn out this:

**Fig. 12**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  | 1 | 1 | 1 |  |  |
| D |  |  |  |  |  |  |  |  |  |
| A |  | 3 |  | 1 |  |  |  |  |  |
| B |  |  | 3 |  |  |  |  | 1 | 1 |
| A |  | 2 |  | 3 |  |  |  |  |  |
| B |  |  | 2 |  |  |  |  | 1 | 1 |

with the similarity magnitude *S*(D1, D2) = 3+3+3+1+1 = 11.

With CHMAXW=3 and fractional weights because of subcommand TRSM (**Fig. 9**) the matching will turn out this:

**Fig. 13**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E | A | B | A | C | C | C | B | B |
| C |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  | 0.2 | 0.2 |
| A |  | 3 |  | 1 |  |  |  |  |  |
| B |  |  | 2.4 |  |  |  |  | 0.8 | 0.8 |
| A |  | 2 |  | 3 |  |  |  |  |  |
| B |  |  | 1.6 |  |  |  |  | 0.8 | 0.8 |

with the similarity magnitude *S*(D1, D2) = 3+3+2.4+0.8+0.2 = 9.4.

In all three cases (**Fig. 11 – Fig. 13**) of the given examples, the pairing elements of **M** (these co-occurrences are marked red) have turned out to be practically the same. In general case, it won’t be necessarily so under different regimes (specifications). Rows and columns of **M** get paired so as to maximize *S*(D1, D2), the sum of pairing co-occurrences. It is the resultant *S* what is of interest, and not which specifically elements became pairing ones. Co-occurrences of greater size (weight *w*) have priority to become pairing, that is, to sum together into *S*.

The macro offers two matching algorithms to select from. Greedy algorithm METHOD=GREEDY approaches openly *hierarchically*: the order of rows/columns pairing goes in the order of considering positive values in **M** from greater to lesser. Not seldom, this manner is precisely what will satisfy the user’s intuition: to accrue the similarity first of all by the weightiest co-occurrences (co-occurrences weights were differentiated in the stage 2 of the algorithm, if that stage wasn’t skipped), and yet then – by what has left to account (i.e., to pair). The slower algorithm METHOD=HUNGAR acts more scrupulously and *guarantees* the maximal possible *S*, reachable with the given matrix **M**, globally. See details – s/c METHOD.

The elements of **M** due to which the pairing took place – the pairing co-occurrences – we’ll call *useful*. Thus, *the raw similarity* of two documents is the sum of *U* useful co-occurrences’ weights:

If there was no stage 2 of the algorithm (altering of weights), all *w* = the initial weights. If there was no substage 2a (CHMAXW=1), we have the document comparison only by content. Along with it, however, substage 2b might have been present, penalizing some *w* in the matrix by their reduction (hence, deflating *S*). If there was substage 2a (CHMAXW not 1), some *w* could have got weights 2, 3, or higher, the this is, to a certain extent, the document comparison by word content and order of adjacent words, i.e., comparison of sequences. Along with it, again, substage 2b might have been present, penalizing some *w* in the matrix by their reduction (hence, deflating *S*). Finally, subcommands TRV/TRSM, which allow weights of co-occurrences to be below 1 even at the moment of creation of **M** matrix, is yet one more, and the earliest in time, source of decrease of *S*. Please see **Fig. 16**.

Raw similarity *S* is the numerator of the similarity coefficient.

*Note*. With method GREEDY, one can request doing, prior the matching, reordering of rows and columns of **M** matrix by the magnitude of row/column marginal sum (keyword REORD). When **M** is highly discrete (has few distinct values) but is not dichotomous, reordering may help greedy algorithm to score a higher *S*, approaching the global maximum.

*Resume.* Maximal matching leaves in the co-occurrence matrix no more than one positive weight in each row and column. The sum of these pairing (useful) weight is the objective function to be maximized; this sum is the raw similarity of the documents.

*Maximal aligning approach*

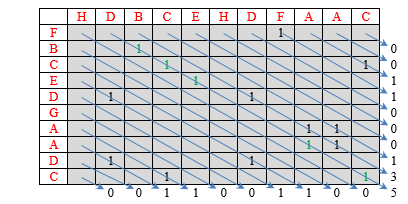
METHOD=MXTRACE. Let sequence D1 be FBCEDGAADC, sequence D2 be HDBCEHDFAAC. If to pull one sequence along the other, each time moving by one position, we’ll eventually find the counter-position (“alignment”) at which the number of same words among the opposite-faced words is maximal for this pair of sequences. Here is the alignment:



It yields 5 coincidences (shown green). And this is the magnitude of similarity of the two sequences per the idea of maximal aligning. Pay attention to the otherness from the idea of maximal matching: there are still remaining co-occurred words in the sequences, for example, D or F. Matching would make use of them: the similarity would be numerically higher (it would reach 9). But alignment takes care of maximizing face-to-face, counter-positioned coincidences only, not the total rate of coincidence. In other terms, maximal aligning selects most weighty, identical in the sequences, distribution of words along them[[5]](#footnote-5).

Having a co-occurrence matrix **M**, the size of maximal alignment is easy to establish. It is the maximal of all the diagonal (slant) sums. Say, in the considered example, the maximal diagonal sum is 5:

**Fig. 14**



Whichever matrix **M** looks like at the time to reckon the similarity, the macro acts standardly – it simply calculates all diagonal sums and chooses the largest of them to be the raw similarity. For example, if **M** at that moment looks as on **Fig. 6**, the similarity will be 6. And if **M** has come up to stage 3 in the form as on **Fig. 10**, the similarity will equal 4.45. In the aligning method of similarity reckoning, all nonzero co-occurrences lying on the diagonal where the sum is maximal – may be called “useful”: and they are summed into the similarity.

*Resume.* Maximal aligning (simple, fixed version) is the principle to establish similarity, which amounts to selecting the maximal sum of all diagonal (slant) sums in the co-occurrence matrix. “Useful” co-occurrences, summing into the raw similarity, lie on one diagonal in the co-occurrence matrix.

*Maximal common chain approach*

METHOD=MXCHW. Possible, if there was substage 2a (award of concurrence chains), i.e., CHMAXW is specified other than 1). Let *wmax* be the maximal weight in **M** at the moment of finishing chains highlight[[6]](#footnote-6) (and *prior* the multiplication by the initial **M** with fractional weights, if TRV/TRSM was specified – see above, “Weighting by chain length in situation of TRV or TRSM subcommands”). Let *lmax* be the length of the longest common chain (chain of concurrence) existing between the two documents. Then the following cases are possible:

**Tab. 1**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | CHMINW ≤ *lmax* | CHMINW > *lmax* |
| CHMAXW=NOLIM | | *wmax* = *lmax*  only chains of length *lmax* are rewarded with weight *wmax* | *wmax* = 1 (if INI keyword) or 0  therefore METHOD=MXCHW simplifies in METHOD=MXW for this pair of documents |
| CHMAXW=*n* | *lmax* < *n* |
| *lmax* ≥ *n* | *wmax* = *n*  all chains of lengths [*n*, *lmax*] are rewarded with weight *wmax* | impossible: CHMINW cannot exceed *n* |

*Note*. If SQUARE keyword in CHMAXW s/c, *wmax* is squared.

Let’s call the common chains, which elements were highlighted with weight *wmax*, “long chains”, set *L*. It reads in the table, that these are the chains of the maximal length *lmax* or, sometimes under CHMAXW=*n*, of the lengths from *n* and up.

Per METHOD=MXCHW, the raw similarity is equal to the sum of weights at entering stage 3 (reckoning raw similarity) of the chain from *L* having the greatest such sum.

So, if **M** looks like **Fig. 14** while entering stage 3, then *S* = 3: the sum of weights in the chain BCE. For **M** on **Fig. 12**, S = 3+3+3 = 9, and for **M** on **Fig. 13**,S = 3+2.4+3 = 8.4. *S* is the weight of most weighty of the “long chains”.

Mark a particular case: if subcommands TRV/TRSM and DIAG were not used, weights in **M** have not changed since the moment of chain highlight in substage 2a, and so the greatest sum in stage 3 will belong to every longest – of length *lmax* – chain: *S* will equal *wmax*∙*lmax*, i.e., similarity *S* is then proportional to the length of the longest chain of concurrence.

*Resume.* Maximal common chain is the principle to establish similarity, which amounts to collecting “long common chains” and then taking the “weightiest” of them as the similarity measure. A specific case: the similarity simply presents the length of the longest common chain. “Useful” co-occurrences, summing into the raw similarity, constitute one concurrence chain.

*Other approaches in the macro*

Other approaches of reckoning the similarity, existing in the macro – are more trivial and need no special explanation. It is possible to take, for the raw similarity, the mean of positive weights in **M** (METHOD=MEANPW), the maximal weight in **M** (METHOD=MXW), or the sum of weights in **M** (METHOD=SUM).

1. **Normalizing into similarity coefficient**

Option to normalize exists in the macro for the methods (approaches) to accrue the similarity: matching, aligning, common chain. These are methods which pair words, i.e., they sum up “useful co-occurrences”.

If DLEN=NONE, the raw similarity *S* obtained on stage 3 is output as the final: *Sim*(D1, D2) = *S*. If *S* = 0, then *Sim*(D1, D2) = 0. In other situations *S* is normalized into the similarity coefficient:

where *DIVISOR* is explained below, and *DLEN* is set by DLEN subcommand and by default equals the length of the smaller of the two documents: . (Meant is the net length, not counting gaps.)

The idea of normalizing is confined in that the denominator equals the maximal possible size of for the given pair of documents D1, D2. Raw similarity *S* is maximal for the given pair when the smaller of the two documents entirely consists of chains, highlighted by weight *DIVISOR* on exit from stage 3. Formally, *Sim*(D1, D2) = 1 (maximal), if:

Let us repeat: you may expect *Sim*(D1, D2) = 1 only when *all* the useful (pairing, summing themselves up into the similarity) elements have weight equal to *DIVISOR*.

*Effect of diagonality*

The formula of similarity coefficient *Sim*(D1, D2) does not compensate for a retreat from the diagonal structure in a co-occurrence matrix, therefore *Sim*(D1, D2) can attain 1 if only the co-occurrence matrix is diagonal. You may omit the DIAG subcommand in order to set aside check of diagonality.

*Effect of relevances (and similitudes)*

The formula of similarity coefficient *Sim*(D1, D2) does not normalize by relevances/similitudes of terms (s/c TRV/TRSM), therefore *Sim*(D1, D2) can attain 1 if only all the words forming the useful co-occurrences in stage 3 possess full relevance/similitude 1. If some terms are under-relevant (but you did not exclude them from the data), this must decrease *Sim*, and there is no need to compensate for that.

*Effect of DLEN and choice of DLEN*

*Sim*(D1, D2) = 1 can only be under or under (documents of equal length). As said before, by default. Subcommand DLEN allows to choose value for *DLEN* (the normalizing document length) other than the default one. For example, you may prefer the length of the bigger document of the two, , or the average length of the two documents. You may refuse doing normalization: DLEN=NONE. Also, you may save information on document lengths, and information on maximal weights observed in **M** after stage 1 or 2a (see s/c SAVEMXAW), in order to use that for working out your own denominator, normalizing *S* into *Sim*.

*Choice of value of DIVISOR*

If there was no highlight of chains (CHMAXW=1, substage 2a skipped), *DIVISOR* = 1. If there was highlight of chains, necessary to decide what to assume this parameter. At normalizing, *DIVISOR* removes the *effect of chain reward* by an increased weight – it takes away weight inflation. That is needed in order the theoretical upper bound of *Sim* be 1, and not higher.

Let *wmax* be the maximal weight in **M** at the moment of finishing chains highlight (and *prior* the multiplication by the initial **M** with fractional weights, if TRV/TRSM was specified – see above, “Weighting by chain length in situation of TRV or TRSM subcommands”)[[7]](#footnote-7). As follows from the description of substage 2a “Weighting by length of concurrence chain”,

where *lmax* is length of the longest chain of concurrence existing between the two given documents.

*Note*. In the presence of keyword SQUARE in CHMAXW, *wmax* is squared.

When there are only two documents in the analysis, a natural choice *DIVISOR = wmax* brings no problem about. But when there are more than two documents in the analysis, using *DIVISOR = wmax* has a consequence which may not suit the user. Here is an example of several **M** matrices (at the moment of finishing chain highlight) selected from the same analysis with several documents:

**Fig. 15**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  | 1 |  |  |  |
|  | 5 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  | 1 |  |  |  |  |  |
|  |  | 5 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  | 1 |  |  |
|  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |
|  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |

**a b c d**

*S* = 25 9 9 5

*DLEN* = 5 5 3 5

*wmax* = 5 3 3 1

Suppose we are using maximal matching as the method to reckon similarity. Then the raw similarity *S* will be as shown under the matrices. The *DLEN* divisor (the smaller document’s length) is everywhere 5, except case **c** where it is 3.

If *DIVISOR =* *wmax*, *Sim* equals: 25/(5∙5) = 1 in **a**, 9/(3∙5) = 0.6 in **b**, 9/(3∙3) = 1 in **c**, 5/(1∙5) = 1 in **d**. While the user may like that in both cases **a** and **c** – where *S* appears maximal possible – *Sim* = 1, they will find that in **d** *Sim* also came 1. This may be not to a user’s liking who undertook chain highlight precisely in order to get similarity in cases **a** and **c** higher than in case **d**. That advantage over **d** was fixed in the value of *S*, but now it got lost at the normalizing into *Sim*. Choice *DIVISOR = wmax* – is the *decision forgetting the weighting of chains*; it converts, at normalization, co-occurrences having been rewarded with an elevated weight back to weight 1. This option exists in the macro as DIVISOR=DIVISOR1.

Observe another choice of *DIVISOR*. Let *DIVISOR = Wmax*:

*Wmax =* max(*wmax\_1*, *wmax\_2, …, wmax\_q*) – the largest of all *q* *wmax* (*q* is the number of document pairs under comparison in the current analysis).

*Wmax* is a constant for the entire current analysis. Suppose *Wmax =* 5 in our example. Then *Sim* equals: 25/(5∙5) = 1 in **a**, 9/(5∙5) = 0.36 in **b**, 9/(5∙3) = 0.6 in **c**, 5/(5∙5) = 0.2 in **d**. User who has utilized chain rewarding will be satisfied with the result in case **d**: the similarity is small, as expected. But now the similarity got downgraded in case **c** either, – this may be not to a user’s liking who considers that in **a** and **c** the normalized similarity should be the same, and maximal. Choice *DIVISOR = Wmax* – is the *decision affirming the weighting of chains*; and a *rigorous* decision. Let’s explain why. When we divide *S* by *Wmax*, we “punish” co-occurrences for them weighting less than *Wmax*. We say, addressing to the case **c**: “here is matrix **a**, wherein weights reached even weight *Wmax* (5), while in you, **c**, they showed only modest success (3)”. But is this reproach to case **c** fair? No, because in case **c** a weight could never reach 5 during chain highlighting: the length of the smaller document, equal to 3 here, does not permit it. In case **b** – there it could be, for the length of the smaller document is 5 there. Therefore, the aforesaid “reproach” is fair in case **b**, but not in case **c**. That’s why *DIVISOR = Wmax* is a “rigorous” decision. This option exists in the macro as DIVISOR=DIVISOR2.

It follows from the previous paragraph, that a *fair decision affirming the weighting of chains* would be *DIVISOR =* min(*Wmax*, *wulimit*), where *wulimit* is the upper limit for *wmax* in the given **M** matrix. As said, this limit is defined by the length of the smaller of the two documents; but it can be even less in case of CHMAXW=*n*:

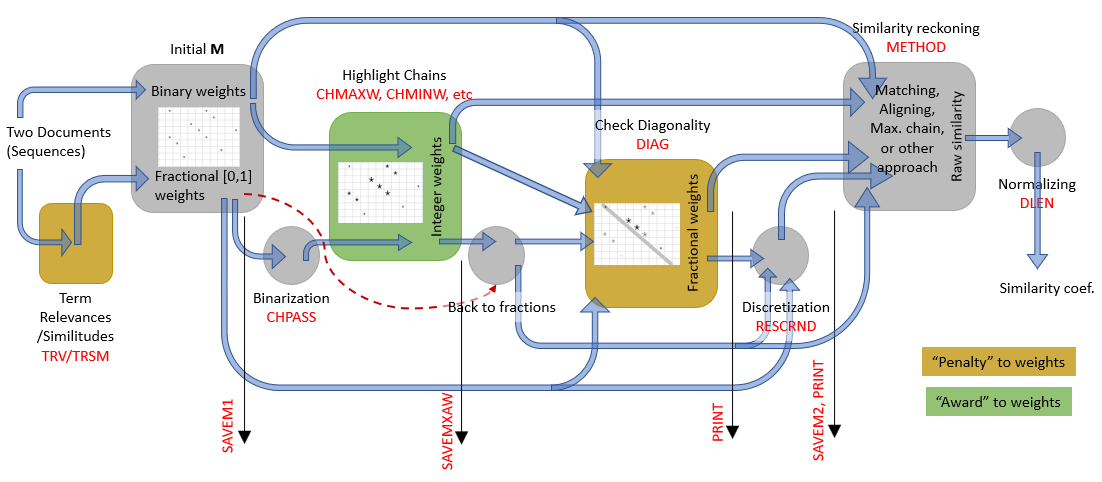
*Note*. In the presence of keyword SQUARE in CHMAXW, the lengths of the documnts |D|, like *n* is, are squared.

If in the given document pair *wulimit* is less than *Wmax*, then this *wulimit* is used instead of *Wmax* in that pair, as the *DIVISOR*. So then *Sim* in case **c** appears maximal again, 9/(3∙3) = 1, while in the three other cases (**a**, **b**, **d**) *Sim* remains the same as it was with *DIVISOR = Wmax*. Option *DIVISOR =* min(*Wmax*, *wulimit*) exists in the macro as DIVISOR=DIVISOR3 and is used by default.

One more variant: *compromise decision affirming the weighting of chains*. It is a “compromise” between the “rigorous” and the “fair” variants: *DIVISOR =* harmonic\_mean(*Wmax*, *wulimit*). Between the two values, *Wmax* and *wulimit*, the harmonic mean is calculated. Then *Sim* in case **c** will be slightly lowered, 9/(3.75∙3) = 0.8 (it is between 1 of the “fair” variant and 0.6 of the “rigorous” variant), and in the three other cases (**a**, **b**, **d**) *Sim* remains – in our example – the same as it was with *DIVISOR = Wmax*. Option *DIVISOR =* harmonic\_mean(*Wmax*, *wulimit*) exists in the macro as DIVISOR=DIVISOR4.

*Note*. If there are just two documents in the analysis, DIVISOR1 = DIVISOR2 = DIVISOR3 = *wmax*.

*Note*. There is an option (subcommand WMAX) to set *Wmax* on a constant value the user wishes, not depending on the data set, that is, on values *wmax*.



**Fig. 16**. Flow chart of the macro’s algorithm.

**Speed performance**

Comparing sequences is time- and resource-consuming (in contrast to mere frequency-based comparison of documents). The macro computes raw similarity *S*(D1, D2), taking the documents in pairs – so, processing time grossly depends on the number of documents. And, naturally, on their size (length). All operations with the current co-occurrence matrix **M** (see “Algorithm”) are done in RAM memory. Being written on a high-level interpreted language – which SPSS syntax is – the macro inevitably is limited in its fluency qualities; due to what it is suited mostly for tasks of small or medium scale.

* METHOD=HUNGAR is much slower than the greedy method of matching and is not recommended if your documents are of thousands of words. METHOD= GREEDY REORD is somewhat slower than METHOD=GREEDY.
* METHOD=MXTRACE (aligning) and METHOD=MXCHW (maximal chain) possess medium speed. Methods SUM, MEANPW, MXW are very fast.
* Specifying DIAG is a considerable source of slow-down. Specifying TRSM/TRV also brings about slowing down. In these cases, the essential cause of slowing of the process of matching (that is METHOD=GREEDY/HUNGAR) is the great diversity of weights in the co-occurrence matrix. To struggle it, use RESCRND s/c.
* PRINT= MIDDLE or LONG bring about slow-down. Don’t use LONG when the documents are long.

***Subcommands***

**DOCVAR**

Indicate short string (up to 8 bytes wide) variable which values are the names of documents (sequences). The values must be already grouped together: dataset cases belonging to the same document must be near. Values must start with a letter or character @ and be a whole word – these document names will become variable names in the dataset that the macro will output. No empty cells or missing values may be in DOCVAR variable.

**WORDVAR**

Indicate string or numeric variable which values are the sequences’ elements (document “words”). Each dataset case is an element, or word. WORDVAR variable will on input be subject by the macro to autorecoding, by SPSS command AUTORECODE, into integer codes 1, 2, 3, ..., and the analysis will be conducted already with those. The autorecode variable – its name is *WORD\_.#$* – will be saved in the input dataset. A next call of the macro overwrites it. Use s/c PRINT (see) to get the autorecode protocol in Output Viewer.

*Gap vs word removal*. A missing value or a blank in WORDVAR variable is interpreted as the “gap”, not a word. It means: document words separated by gap won’t be able to play as adjacent, as members of a chain. A gap is like a word dividing neighbouring words and itself not participating in the analysis anyhow, it doesn’t even enter the length of the document. Presence or absence of gaps matters only when CHMAXW parameter is not 1, i.e., when the issue of succession of adjacent words is raised. If CHMAXW=1 (which is the default), a gap is equivalent to deletion (or filtering off) of a word from input data. Deletion of a word links the words surrounding it, making them adjacent, however under CHMAXW=1 the issue of adjacency, of chains, simply does not arise. About setting zero relevance for a term (one more option) – see in s/c TROTR.

**COMPARE**

Set the work for the macro:

ALLPAIRS - (default/unspecifying) return square symmetric similarity matrix between all the documents, pairwise.

FIRST - return similarity vector between the 1st document and each of the other documents.

FIRSTEXT - extract the longest chain (continuous subsequences of elements) of words coinciding between the documents. One chain is extracted from the comparison of the 1st document with each of the other documents. This option ignores subcommands CHMAXW/CHMINW, CHBWARD, TRV/TRSM/TROTR, DIAG/DIAGLS, RESCRND, METHOD, DLEN/DIVISOR/WMAX, PRINT, because it does not compute the similarity measure. COMPARE=FIRSTEXT requires s/c DATASET (see) be specified.

COMPARE=FIRSTEXT extracts (shows) just one longest chain of concurrence from each comparison. If there are more than one chain of the same length, the macro informs of that. You can see all concurrence chains existing between two documents, using s/c SAVEM1.

EXAMPLE 2. Search query.

data list free /content (a8).

begin data

A B C D E F G H

C B M A D B B M L J A D C J

C B L N B C D E F K D M M C D

A K A A B C D E F G H J J A A C D L M K A B C D E F G H H

E A B C D E F G J C D E F G H A K F

end data.

string seq (a8).

do if $casenum<9.

compute seq= 'seq0'.

else if $casenum<23.

compute seq= 'seq1'.

else if $casenum<38.

compute seq= 'seq2'.

else if $casenum<67.

compute seq= 'seq3'.

else.

compute seq= 'seq4'.

end if.

execute.

dataset name data.

* Data entered.

!KO\_seqsim docvar= seq /wordvar= content /compare= FIRST /chmaxw= 8 /chminw= 8 /dlen= NONE

/method= SUM.

* COMPARE=FIRST compares the 1st document with every other. This suits the task “search query”.
* Here, the first sequence from *SEQ* (its name is seq0) is compared with each of the rest. The researcher made the first sequence to be his search query: it all is the sought-for phrase ‘A B C D E F G H’ of 8 words in the needed order, the researcher wants to know if the sequence encounters in full inside the other, more lengthy documents.
* CHMAXW=8 means that an encounter (in the other documents) of this chain of 8 words will be 8-time rewarded, while CHMINW=8 means that any partial concurrences, i.e. shorter than 8 words, won't be paid attention. Consequently, the raw (not normalized by document lengths) similarity between the search sequence seq0 and each other document can, per METHOD=SUM (just the sum of elements in the co-occurrence matrix) be: either *k*∙64 (found), or 0 (not found). *k* is the number of instances of the subsequence found.
* In our example, similarity between seq0 (the query) and seq3 is equal to 128 – i.e., in the document seq3 there present are two instances of the sequence ‘A B C D E F G H’.

!KO\_seqsim docvar= seq /wordvar= content /compare= FIRST /chmaxw= 6 /chminw= 6 /dlen= NONE

/method= SUM /print= LONG.

* In these two runs with the same data and the same search query (seq0) the researcher got interested in any concurrences of length not less than 6 going in succession words of those 8 words. So he lowered the demands, parameters CHMAXW and CHMINW, to 6. Now the similarity size 48 (i.e. 6∙8) will be the sign of presence of the entire chain of 8 words in a document, similarity 42 (6∙7) reveals concurrence by some 7 going in succession words out of 8, and similarity 36 (6∙6) – concurrence by some 6 going in succession words out of 8. So will be if a chain is found in one instance.
* Let us see what we’ve got in our example. For better orienting in the results, s/c PRINT=LONG printed out co-occurrence matrices for us. Between seq0 (the query) and seq3 the similarity METHOD=SUM is equal to 96. This is because in seq3 there are two instances of the sequence ‘A B C D E F G H’, each “weighting” 6∙8. Between seq0 and seq4 the similarity is equal to 78. This sum of the weights is 6∙7 + 6∙6: for, in seq4, two instances of incomplete concurrence with the search query were found: one 7 words length (namely, ‘A B C D E F G’), and the other is of 6 words (‘C D E F G H’).

**TRV**

You will need subcommands TRV/TRSM and TROTR in the case when you want to give different words different importance in the act of comparing documents and computing the similarity between them. TRSM additionally allows to designate word synonymy. TRV/TRSM are in effect on the stage of creating the initial co-occurrence matrix (see “Algorithms”) and affects subsequent stages.

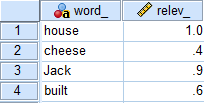
Optional subcommand TRV (“term relevance vector”) allows to give distinct words – terms – different relevances (importances) in the act of document comparison. Relevance is a number in the range [0,1], where 0 means “the term is entirely unimportant in the task of document comparison” and 1 means “the term is fully important in the task of document comparison”. By default/unspecifying the subcommand, relevance of any term is taken for 1, which means: all the words of the input documents are equally and maximally important for document comparison. You set different relevances to terms by TRV subcommand or by TRSM subcommand (you are in the right to specify only one of these two subcommands).

In TRV, indicate one of the two:

AUTO - the macro will compute relevance of every term as , where *d* is the number of documents (of all the corpus of documents input, according to DOCVAR variable) in which the given term encounters. Therefore, a term occurring only in few of all the documents being analyzed by the macro becomes more important for pairwise comparison of the documents than a term occurring in many of the documents. A term occurring only in one document will receive importance (relevance) 0.

*file/dataset* - relevances for terms are specified by the user, by indicating an external .SAV file or open dataset that is the reference on term importance. *Path/name* of a file in quotes or apostrophes or dataset *name*. Terms existing in the data but absent in the reference will be appointed relevance TROTR (see the subcommand, 1 by default).

The reference file/dataset must consist of variables *WORD\_* and *RELEV\_*. The first is string or numeric, corresponds to the WORDVAR variable entering the macro, and contains the list of words (terms) which you want to give individual relevance; the second variable is the relevances themselves, numbers from 0 to 1. If WORDVAR variable is string, *WORD\_*, also string, must have the same width as it. Missings in *RELEV\_* variable evaluate to absence: such terms will be appointed relevance TROTR. An example of reference file for EXAMPLE 1:



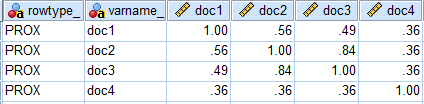
*Giving zero relevance to a term*. See discussion of the issue in s/c TROTR.

Since relevances below 1, i.e., non-full importance, are coefficients decreasing the magnitude of similarity, it is pointless to compare directly similarities obtained with TRV s/c specified with results obtained without it.

TRV creates, in the input dataset, variable *RELEV\_.#$*, showing relevances of the words. TRV=AUTO, besides, creates variable *NSEQW\_.#* showing the number of documents in which the word occurs.

EXAMPLE 3. Use EXAMPLE 1 data.

!KO\_seqsim docvar= doc /wordvar= word /trv= relev /method= GREEDY.



* *RELEV* is an open and not active dataset, containing relevances for terms **house**, **cheese**, **Jack**, **build** (see the picture of it above earlier); while for the rest terms relevance, by default =1. The rest of specification is the same as it was in the 1st run of EXAMPLE 1. Relevance below 1 decreases the size of similarity.

**TRSM**

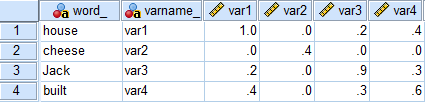
Optional subcommand TRSM (“term relevance & similitude [=similarity] matrix”) is an extended version of the TRV s/c. TRSM allows to give distinct words – terms – different relevances (importances) in the act of document comparison, and also to specify the degree of similitude between different terms. For a user, a high similitude of two terms may bear subjective meaning of synonymy (the terms are exchangeable) or contextual affinity (the terms often meet together in common-theme documents).

Both relevance and similitude – are numbers in the range [0,1], where 0 means zero importance (or similitude) and 1 means full importance (or similitude). Relevance and similitude are concepts in the same line, juxtaposed: relevance is the importance of co-occurrence of the term with own self, and similitude is the importance of co-occurrence of two different terms. In either case the importance of co-occurrence plays “for” likeness of the being compared documents, if it is close to 1, and plays “against” their likeness, if it is close to 0.

TRV subcommand allows to input relevances, in the form of a vector. TRSM subcommand allows to input both relevances and similitudes, - it is a square symmetric matrix. You are in the right to specify only one of these two subcommands. By default/unspecifying the subcommands, relevance of any term is taken for 1, which means: all the words of the input documents are equally and maximally important for document comparison; and similitudes for all terms is taken for 0, which means that different terms are not associated a priori: they are *different*.

In TRSM, indicate an external reference .SAV file – *path/name* in quotes or apostrophes. Terms existing in the data but absent in the reference will be appointed relevance TROTR (see the subcommand, 1 by default). As for similitude value for terms absent in the reference, it is taken for 0, fixed.

The reference file must begin with variable *WORD\_*. It is string or numeric, corresponds to the WORDVAR variable entering the macro, and contains the list of words (terms) which you want to give individual relevance/similitude. If WORDVAR variable is string, *WORD\_*, also string, must have the same width as it. The second must go short (up to 8 bytes wide) string variable *VARNAME*\_ with the names of the variables forming the matrix body. Next, the body of the square symmetric matrix must go. Term relevances are on its diagonal. Similitudes between different terms are its off-diagonal entries. No missing values should be in the file. No extraneous variables should be in it. An example of reference file for EXAMPLE 1:



Typically, similitudes (off-diagonal values) in your matrix will be lower than relevances (diagonal values), because, as a rule, presence of ‘A’ and ‘A’ in both documents influences the document similarity stronger than presence of ‘A’ in one and ‘B’ in the other. But, speaking generally, that is not a requirement.

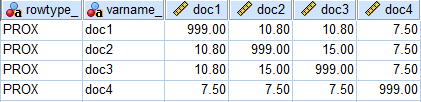
If all off-diagonal elements are zero, the effect is the same as when to enter just diagonal elements as the relevance vector via TRV subcommand (but the latter will be advantageous from the speed performance point of view).

TRSM demands s/c DATASET be specified (see).

EXAMPLE 4. Use EXAMPLE 1 data.

!KO\_seqsim docvar= doc /wordvar= word /trsm= 'd:\exercise\trsm.sav' /trotr= .7 /chpass= .3 .3

/chmaxw= 3 /method= HUNGAR /dlen= NONE /dataset= data.



* *TRSM* is a file, the matrix containing relevances and similitudes for the terms **house**, **cheese**, **Jack**, **build** (see the picture of the matrix above earlier); and for the rest terms relevance is set to 0.7 (s/c TROTR). Because reward of common subsequences (CHMAXW=3) is requested, it is necessary to indicate a pass score for admission into the reward procedure: it is CHPASS s/c, the 1st value concerns relevances, the 2nd one – similitudes. In this instance, both thresholds are the same, 0.3. Because similitude between **house** and **cheese** (0.2) is below the threshold, similitude between these two words will be taken for 0 and will be incapable to participate in the document similarity reckoning.
* The rest of the specification is as in the 2nd run of EXAMPLE 1.

**TROTR**

This subcommand (“term relevance, other”) is in effect if TRSM is specified, or TRV is specified as a file/dataset. It appoints relevance to every “other” term – a term not mentioned in the reference file. Specify number in the range [0,1]. By default, it is 1, i.e., full-value importance in the act of comparison. TROTR concerns only relevances, i.e., the weights of co-occurrences of terms with own selves.

*Giving zero relevance to a term*. If you specify relevance 0 for some terms – either through TROTR or in TRV vector or in TRSM matrix, – then the question may be rised, if it is equivalent to deletion or recoding into missing (“the gaps”) those words in the data before the macro run. (We’ll put it for now that, in case TRSM is specified, all similitudes there =0, i.e., the TRSM is equivalent to TRV.) The answer is as follows. For reckoning of *raw* similarity, setting a term’s relevance to 0 is equivalent to its recoding into a gap (and if CHMAXW=1, then also equivalent to deletion of the term from the data). However, for obtaining *normalized* similarity (see DLEN s/c), these are not quite equivalent things, because a word with relevance 0 is included in the document length, but a gap, like a word deleted, is not included in the document length. For the difference between gap and deletion – see in WORDVAR s/c.

**CHMAXW**

You will need subcommands CHMAXW, CHMINW, CHBWARD, and CHPASS, if you want to compare the documents as sequences of contiguous words rather than simply collections of words.

CHMAXW (“chain maximal weight”) is the principal of these 4 subcommands. CHMAXW assigns the maximal rewarding weight to co-occurrences forming continuous subsequences (chains of adjacent words) that concur in the two being compared sequences (documents). Indicate integer *number* 1 or higher or keyword NOLIM (“no limit”). See illustration in “Algorithm: Weighting by length of concurrence chain”.

By default/unspecifying of the subcommand, CHMAXW=1, which means that concurrence chains won’t be valued higher than coincidences of individual, separate words. This meets the task to compare the documents only by word *content*. CHMAXW greater than 1 is already a comparison by *content and order of adjacent* words. The higher is CHMAXW, the more differentially concurrence chains of different length are valued. This parameter is directly linked with the chain length. Say, with CHMAXW=4, chains of length 4+ words, identical between the two documents being compared, will receive weight 4 (each word of such chain); chains of length 3, identical between the documents, will receive weight 3; chains of length 2 will receive weight 2. Weight 1 will remain at words, coinciding between the two documents, which do not enter the chains of concurrence. With CHMAXW=2, chains of length 2+ will receive weight 2. Weight 1 will remain at coinciding words which do not enter the chains of concurrence. So, CHMAXW=*number* – is the limiting reward weight for an element of a concurrence chain.

With CHMAXW=NOLIM, the limit is not put, and the weight of an element of a concurrence chain will always be equal to the observed length of the chain, whatever long it might occur.

Optional added keyword SQUARE makes weights squared, rather than linear. For example, CHMAXW= 4 SQUARE creates weights 1, 4, 9, 16 instead of weights 1, 2, 3, 4, which rises the value of lengthier chains of concurrence in accelerated fashion. Keyword SQUARE may be used after CHMAXW=NOLIM too.

You can use s/c SAVEMXAW (see) to save the maximal observed, after the highlight (reward) of chains, weight in each pair of documents compared.

*Rewarding concurrence chains under TRV or TRSM specified*. When you didn’t specify TRV/TRSM, the initial co-occurrence matrix is binary: weight 1 = co-occurrence (coincidence), weight 0 = non-co-occurrence. The effect of reward (highlight) of concurrence chains by CHMAXW subcommand (when it is set not to 1) is the differentiation of weight 1: in place of weights 1, weights 1, 2, 3,… will occur. While if you did specify TRV/TRSM, the initial co-occurrence matrix is likely to be non-binary, consisting of various fractional values from 0 to 1. In order CHMAXW subcommand to be able to act, i.e., to reward chains of units, it is necessary first to dichotomize the fractional values of the matrix into binary: 1 and 0. Subcommand CHPASS (see) manages it.

EXAMPLE 5.

set rng= mc seed= 4056748. /\*You may set another seed

input program.

string seq (a8).

leave seq.

loop #seq= 1 to 10. /\*Let there be, say, 10 sequences (documents)

compute seq= concat('seq',ltrim(string(#seq,f3))).

loop #case= 1 to rnd(rv.uniform(50,100)). /\*Each containing, say, 50 to 100 words (cases)

compute word= rnd(rv.uniform(1,8)). /\*Let a "word" be an integer from, say, 1-8 range

end case.

end loop.

end loop.

end file.

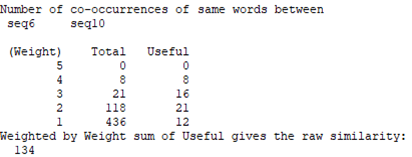
end input program.

execute.

dataset name data.

* Some generated data.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /method= GREEDY PRINT.



* Sequences are compared in pairs and the similarity matrix is produced.
* The researcher is interested not only in word composition, but also in the order of contiguous words, because CHMAXW is specified and is not equal to 1. CHMAXW=5 awards concurrences (coincidences) of length 5+ of adjacent words with weight 5, concurrences of length 4 of adjacent words with weight 4, concurrences of length 3 of adjacent words with weight 3, etc.
* METHOD=GREEDY, doing the pairing (matching), counts coincidences, starting from most weighty. Since possible weights in this example are integer (1, 2, 3, 4, 5), you may use keyword PRINT to report on the coincidence weights.
* In this case we see, that, for example, between documents seq6 and seq10 there is not a single common chain of length 5, there are two common chains of length 4 (therefore the sum of weights of these co-occurrences is 8), there are 7 common chains of length 3 (therefore the sum of weights 21), and so on. During the pairing, there were accounted (i.e., built up the similarity) all 8 most weighty co-occurrences, 16 co-occurrences of weight 3, 21 co-occurrences of weight 2, etc.

!KO\_seqsim docvar= seq /wordvar= word /trv= AUTO /chpass= .1 /chmaxw= 5 /method= GREEDY.

* In this run, unlike the previous, words (terms) have unequal relevance (importance in the act of document comparison): TRV and CHPASS are specified. Similarity magnitude between all the documents will decrease, because relevances will come to be less than 1 (in this example, they all are the same and equal 0.301, though usually relevances are of interest when they are different).
* Co-occurrence weights (because of TRV) are fractional, therefore keyword PRINT was removed.

**CHMINW**

This subcommand (“chain minimal weight”) assigns the minimal rewarding weight to co-occurrences forming continuous subsequences (chains of adjacent words) that concur in the two being compared sequences (documents). Indicate integer not less than 1. The effect of the subcommand is as follows. *After* the CHMAXW will reward chains of concurrence with an elevated weight, CHMINW will erase (zero) all co-occurrences whose weights appeared to be less than CHMINW value. Adding keyword INI after the number will make the subcommand not to erase all such co-occurrences, but rather leave them their initial (pre-reward) weight, i.e. 1. Thus, INI withdraws not short chains from circulation, it retains them having the same weight as co-occurrences outside chains have.

For example, CHMINW=4 will annihilate all co-occurrences forming concurrence chains of length 3, 2, and 1 (1 means co-occurrences not forming chains). But CHMINW= 4 INI will give all these co-occurrences weight 1 (i.e., will return them their initial weight). CHMINW=2 will annihilate all co-occurrences of weight 1, i.e., not forming chains. But CHMINW= 2 INI will keep them, because will return them their initial weight 1 (which they never changed). Note that CHMINW= 2 INI is equivalent in result with CHMINW=1.

By default, CHMINW=1. That means that individual words, identical in the two documents, are accounted in the establishment of document similarity. Raise CHMINW above unit if you are interested in similarity due only to chains of length not less than so many, rather than due to separate words. Raise CHMINW above unit and supply keyword INI, if you are interested in similarity mainly due to chains of length not less than so many.

Value in CHMINW is indicated in linear scale, as in CHMAXW. If keyword SQUARE is added in CHMAXW, it automatically expands over to CHMINW parameter. Thus, in this case CHMINW=3 will be interpreted as the minimal weight 9.

When CHMAXW=NOLIM and the CHMINW value specified by you appeared greater than the length of the longest chain existing in that pair of documents, all chains will occur zeroed (if you don’t add keyword INI) and the similarity will be 0.

When CHMAXW=*number*, CHMINW cannot be specified by you greater than that number CHMAXW. If the longest chain existing in that pair of documents is shorter not only than CHMAXW, but is shorter than CHMINW, all chains will occur zeroed (if you don’t add keyword INI) and the similarity will be 0.

EXAMPLE 6. Using EXAMPLE 5 data.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /chminw= 3 INI /method= GREEDY.

* Unlike the 1st run in EXAMPLE 5, CHMINW= 3 INI is added. That means, the macro will award concurrences of length 5+ of adjacent words with weight 5, concurrences of length 4 of adjacent words with weight 4, concurrences of length 3 of adjacent words with weight 3, but concurrences of length 2 - it will award only with weight 1, like single coincidences (of length 1).

**CHBWARD**

This subcommand (“backward chains”) plays a role if CHMAXW is not 1. By default/unspecifying and with CHBWARD=NO, concurrence chains only with identical, direct order of words are rewarded with elevated weight. With CHBWARD=YES, chains with reverse order also are considered concurrent and are rewarded with elevated weight. For example, the line of words ‘A B C’ in a document will count as a concurrence chain not only with the same line of words in the other document, but also with the line ‘C B A’ in it. To put it another way, CHBWARD subcommand makes the macro to count mirror-identical subsequences identical too, like co-identical.

**CHPASS**

This subcommand (“pass score to chains”) acts and is necessary if you specified TRV or TRSM, while CHMAXW is not 1 – i.e., when rewarding of common chains should be. When you didn’t specify TRV/TRSM, the initial co-occurrence matrix is binary: weight 1 = co-occurrence (coincidence), weight 0 = non-co-occurrence. The effect of reward of concurrence chains by CHMAXW subcommand is the differentiation of weight 1: in place of weights 1, weights 1, 2, 3,… will occur. While if you specified TRV/TRSM, the initial co-occurrence matrix is likely to be non-binary, consisting of various fractional values from 0 to 1. In order CHMAXW subcommand to be able to act, i.e., to reward chains of units, it is necessary first to dichotomize the fractional values of the matrix into binary: 1 and 0. CHPASS subcommand manages it, it assigns pass score for taking a part in chains highlighting.

If TRV s/c is specified, the initial co-occurrence matrix contains, besides zeros, the relevances – they act as the weights of co-occurrences of terms with own selves. Indicate in CHPASS one value in the range (0,1]. Term co-occurrences with own selves having weight not less than this value will be turned in 1, and having weight less – will be turned in 0. After such dichotomization, the matrix is submitted to chain rewarding.

If TRSM s/c is specified, the initial co-occurrence matrix contains, besides zeros, the relevances – they act as the weights of co-occurrences of terms with own selves, and also contains similitudes – they act as the weights of co-occurrences between some different terms. Indicate in CHPASS two values in the range (0,1]. The first value concerns co-occurrences of terms with own selves. Such co-occurrences having weight not less than this value will be turned in 1, and having weight less – will be turned in 0. The second value concerns co-occurrences between different terms. Such co-occurrences having weight not less than this value will be turned in 1, and having weight less – will be turned in 0. After such dichotomization, the matrix is submitted to chain rewarding. So, a similitude turned into 1 may become a link of a concurrence chain. Thus, ‘A B C D’ in a document and ‘A B Y D’ in the other document will be deemed coinciding chains, if ‘C’ and ‘Y’ have the similitude between them ≥ pass score for similitudes.

If you specified TRV and two values in CHPASS, the second value is silently ignored.

After the rewarding (highlighting) of chains is done (i.e., all the specified of the subcommands CHMAXW, CHMINW, CHBWARD have played), CHPASS will “play back” the dichotomization it had done, by simply multiplying elementwise the obtained integer-valued co-occurrence matrix by the initial co-occurrence matrix. For example, some co-occurrence, an element of the co-occurrence matrix, initially had weight 0.6. This value happened to be above the pass score specified in CHPASS, so then it turns in 1 (and not 0). Then highlighting of chains found out this co-occurrence belongs to a chain of concurrence of such a length that its elements are worthy to be rewarded with weight 3. In the result: 3∙0.6=1.8 – this weight the co-occurrence will have upon exit from the “rewarding of subsequences” block. While if the initial weight of the co-occurrence was, let’s say, 0.2 and it happened to be below the pass score, it turns into 0 and does not participate in chain rewarding. Since 0∙0.2=0, such co-occurrence from now on is unable to vote for the similarity of the two current documents.

Let us stress: the admission regime CHPASS is in effect only if CHMAXW is specified and is not 1.

EXAMPLE 7. Using EXAMPLE 5 data. Plus you will need to prepare a file with relevances *RELEV.SAV*, file with relevances/similitudes *TRSM.SAV*.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /trv= 'd:\exercise\relev.sav' /chpass= .1

/method= GREEDY.

* In the 2nd run of EXAMPLE 5, TRV=AUTO was used. In the current example, relevances are specified as an external file. And because CHMAXW is not equal to 1, CHPASS is necessary. Every word with relevance 0.1 or higher will participate in chain highlighting by CHMAXW subcommand. After highlighting, the obtained integer weights will be multiplied by the corresponding initial weights taken from the TRV file.
* Subcommand TROTR (accompanying TRV or TRSM) is omitted. By its default, relevance for every term not mentioned in the file =1.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /trsm= 'd:\exercise\trsm.sav' /chpass= .1 .8

/method= GREEDY /dataset= data.

* In this run, unlike the previous, we are setting not only relevances, but similitudes as well. CHPASS requires two parameters – the first for relevances, the second for similitudes. Terms having the degree of synonymy between them 0.8 or higher will participate in chain highlighting by CHMAXW subcommand: they will be considered at this substage as full synonyms. After highlighting, the obtained integer weights will be multiplied by the corresponding initial weights taken from the TRSM file.
* Subcommand TROTR (accompanying TRV or TRSM) is omitted. By its default, relevance for every term not mentioned in the file =1 (and similitudes for not mentioned terms always =0).

**DIAG**

Independent of whether sequences of contiguous words are important to you in determining document similarity or only word composition is important to you in that, you might be bothered by the question whether the fragments (chains, separate words) coinciding in the documents occupy the same or different position in them. In other words, is the co-occurrence matrix close to or far away from the diagonal pattern.

Optional subcommand DIAG is to take the degree of diagonality of the co-occurrence matrix into account. Co-occurrence matrix is maximally close to diagonality when the documents are close to being each other copies. DIAG subcommand will penalize, decreasing their weight, co-occurrences in those rows and columns of the matrix where placement of co-occurrences violates the diagonal ideal. This penalty lowers the similarity between the documents. See “Algorithm: Check of diagonality”.

You may want to apply DIAG in case the documents are similar for you only when words or subsequences (“phrases”) identical between them occupy about the same positions in them. I.e., for instance, ‘A B C D X Y Z’ and ‘X Y Z A B C D’ won’t suit you, because ‘A B C D’ and ‘X Y Z’, albeit common in the two sequences, occupy different positions, their placement diverges. Unlike CHMAXW subcommand, s/c DIAG is not busy with specifically *adjacent* words, but rather considers co-orderedness on the “upper level” of sequences.

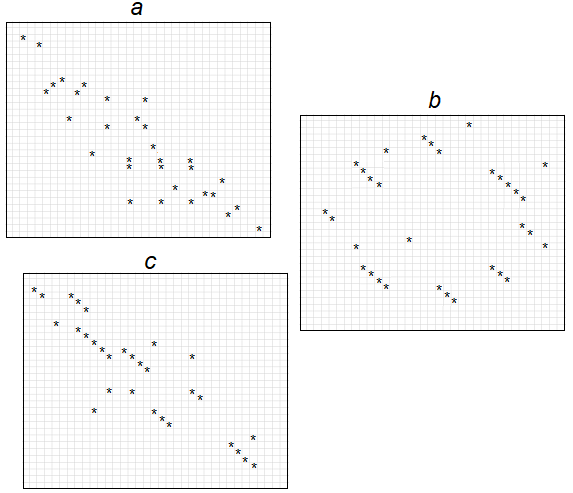
Specify DIAG= MAIN, MIDDLE, or BAND. The difference between these variants emerges when the co-occurrence matrix is nonsquare, i.e., the documents are of different length. MAIN accepts, for the reference diagonal, only the main diagonal, MIDDLE – the middle diagonal, and BAND – all the band of diagonals. See illustration in “Algorithm: Check of diagonality”.

Specifying DIAG is incompatible with CHBWARD=YES.

EXAMPLE 8.

On **Fig. 17**, three different co-occurrence matrices are displayed, each between some two documents. Co-occurrences of identical words between this and that document are shown as points. (Data corresponding to these pictures are found in the “Appendix”.) In case (**a**), the pattern is quite diagonal, while concurrence chains are almost absent. We may comment that the documents are next to be “relaxed” – without fixing the order of adjacent words – copies of each other. In case (**b**), conversely, we see many chains of concurrence, that is, identical “phrases”, however they occupy different positions in the two documents, so the pattern is very non-diagonal. In case (**c**), there both diagonality and chains of adjacent words are present. Here, the two documents are nearly copies of each other on both levels: on the “macrolevel” of placement of parts, and on the “microlevel” of adjoining words.

**Fig. 17**



!KO\_seqsim docvar= seq /wordvar= word /diag= MIDDLE /method= GREEDY.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 4 /method= GREEDY.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 4 /diag= MIDDLE /method= GREEDY.

* The 1st run will “punish”, lowering the similarity, the case (b) – because of s/c DIAG present; cases (a) and (c) will yield rather high and about the same similarity.
* The 2nd run will yield rather high and about the same similarity in cases (b) and (c), and considerably lesser similarity in case (a).
* The 3rd run combines CHMAXW and DIAG, and will yield high similarity in case (c), both other cases will be “punished”, each for its own reason. In this example, case (b) will show similarity roughly half-way between (c) and (b). You can strengthen (or attenuate) “punishment” for non-diagonality, adding DIAGLS subcommand, which is a tuning. Say, if you specify /DIAGLS= 0.35 1 (while the default corresponds to /DIAGLS= 0.5 1), case (b) will be “punished” stronger and will show similarity of about the level of case (a).

**DIAGLS**

This optional subcommand serves s/c DIAG and it is for fine tuning of the diagonality check. By default, the dependence of “penalty” for the size of deviation from diagonal position – is linear. You can make it curvilinear, strengthen of weaken its “harshness”. Specify two values: the first is the location parameter – number in (0,1) range; the second is the shape or steepness parameter – number ≥ 1. See “Algorithm: Check of diagonality”.

**RESCRND**

This optional subcommand acts if DIAG or TRV/TRSM is specified. In these cases, values in the co-occurrence matrix come up to the main stage – the stage of reckoning similarity – in fractional form, as a rule, and diversity of values in the matrix may be large. When the diversity is large, i.e., the data are quite continuous, more time is spent on the maximal matching procedure (METHOD= GREEDY or HUNGAR). RESCRND discretizes the data by the rescale-round way, and this will speed up the matching procedure. See “Algorithm: Rounding co-occurrence weight”. Of course, discretization somewhat roughens, makes less precise calculation of the resultant similarity; however, in the majority of cases it is a tolerable loss, unlike the slow work of the macro.

Indicate number 1, 2, or 3. This is the precision, the number of decimal digits being left at the rounding after the rescaling into the range 0-1. RESCRND=1 is the most rough and fast variant. Unspecifying/omitting of RESCRND subcommand – the most slow and precise variant. RESCRND=2 is often the variant of choice if speed is important to you: acceleration one and a half – two times (with METHOD=HUNGAR) and four – five times (with METHOD=GREEDY) is possible, loss of precision being tolerable.

Methods other than GREEDY or HUNGAR never need s/c RESCRND to be applied, because their speed is independent of the diversity of weights in the matrix. However, their results are of course dependent on whether you applied RESCRND or not.

Use PRINT=MEDIUM (see) in order to know how RESCRND has diminished diversity of weights of co-occurrences.

EXAMPLE 9. Using EXAMPLE 5 data. Plus you will need to prepare a file with relevances *RELEV.SAV*, file with relevances/similitudes *TRSM.SAV*.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /trsm= 'd:\exercise\trsm.sav' /diag= MAIN

/rescrnd= 2 /method= GREEDY /dataset= data.

* Without RESCRND, specification in this run will not be nice for performance, especially if documents are plenty or they are lengthy. Because CHMAXW, TRSM, DIAG are present. Each of these subcommands diversifies weights in co-occurrence matrix, especially that these three subcommands are together here. RESCRND=2 is a remedy in such situation. Not large loss of precision of computations is compensated by a good saving of time.
* RESCRND decreases the fractionizing effect of DIAG and TRSM (or TRV). When these subcommands are absent, RESCRND is ignored.

**METHOD**

The method of obtaining a magnitude of the (raw) similarity of two documents, from the co-occurrence matrix undergone the preceding processing requested by you. Suggested are two variants of the method of maximal matching, method of maximal aligning (basic version), method of the maximal common chain, and some simple methods. For details - see “Algorithm: reckoning raw similarity”. See also **Fig. 2**.

GREEDY - simple greedy method of matching. It is stepwise and (if weights of the elements, i.e., of co-occurrences, are different) begins matching due to most weighty co-occurrences. It always will choose to pair by a weightier element. This method pursues but does not guarantee attaining the global maximum of the matching, that is, revealing the magnitude of raw similarity to its detectable limit.

HUNGAR - Hungarian (Kuhn–Munkres) matching algorithm. It is iterative and maximizes the matching to its global maximum, that is, reveals the magnitude of raw similarity to its detectable limit. It will not always choose to pair by a weightier element over a less weighty element. It is a slower method.

MXTRACE - maximal aligning approach. In the co-occurrence matrix, the sum in each its diagonal is computed (individual diagonal starts from every position of the 1st row or from every position of the 1st column). The greatest such sum is the raw similarity. Unlike matching, there is no goal to include, hopefully, as many co-occurrences (nonzero weights) containing in the matrix as possible, into the similarity. And unlike the MXCHW method, here weights more than of one chain may sum up.

MXCHW - maximal common chain: similarity magnitude is the weight of one, the weightiest among long common subsequences. If s/c TRV/TRSM/DIAG were not used, this is the weight directly representing the length of the longest common subsequence. If any of s/c TRV/TRSM/DIAG were used and CHMAXW=*number*, then the weightiest one may appear not the longest chain. Unlike all the previous methods, in MXCHW method weights of co-occurrences constituting just a single chain are summed into the similarity magnitude. Details – see “Algorithm: Maximal common chain approach”. Method MXCHW is allowed only if you requested weighting of concurrence chains, i.e., specified CHMAXW not as 1. MXCHW is incompatible with CHBWARD=YES.

MXW - similarity magnitude is the maximal weight in the co-occurrence matrix.

MEANPW - similarity magnitude is the averaged nonzero (positive) weight in the co-occurrence matrix.

SUM - similarity magnitude is the sum of all weights in the co-occurrence matrix.

Of the two maximal matching methods, method GREEDY may be considered as preferable in practice. First, it is much faster than the Hungarian. Second, in a condition of sparse matrix of co-occurrence elements (what usually takes place) a local optimum reached by the greedy approach often differs little or even coincides with the global optimum of the Hungarian approach. Third, the preference of a weightier element to a lighter one – forced by the greedy method, may be clearer and more important to a user than the Hungarian algorithm’s interest to “exhaust ultimately” the present similarity to its algebraic limit.

After METHOD=GREEDY, additional keywords REORD and/or PRINT are possible.

Keyword REORD changes, before the matching act, the order of rows/columns in the co-occurrence matrix so as to enhance the chances of the greedy algorithm to approach the global maximum of matching: METHOD= GREEDY REORD maximizes matching usually stronger (closer to HUNGAR) than METHOD=GREEDY. This advantage is maximal when the diversity of weights in the co-occurrence matrix is small, what is realized, for instance, when CHMAXW parameter is low (but greater than 1) and when the manifold of words in the input documents is not very small. Similarity after GREEDY REORD may then be approximately 2-3% higher than after GREEDY. As for CHMAXW=1 in the absence of TRV/TRSM and DIAG, the matrix is dichotomous then: co-occurrences possess just one weight; in these conditions REORD gives no advantage. The REORD reordering of rows/columns spends time on itself, of course.

Keyword PRINT (see EXAMPLE 5) prints out in the output Viewer the summary about the counts of co-occurrences of different weights – before the greedy matching (column “Total”) and after it (column “Useful”). The raw similarity calculated is the weighted, by the weights, sum in “Useful”. See “Algorithm: Maximal matching approach”. Keyword PRINT is not allowed in the following cases: DIAG is specified, TRV/TRSM is specified, CHMAXW=NOLIM, CHMINW contains INI.

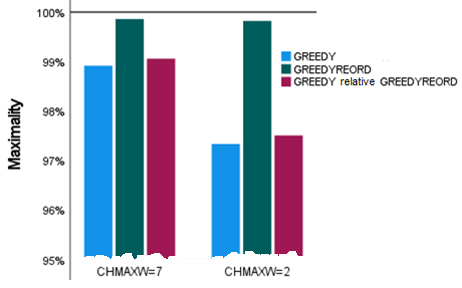
EXAMPLE 10. Using EXAMPLE 5 data.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 7 /method= GREEDY /dlen= NONE /print= MEDIUM.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 7 /method= GREEDY REORD /dlen= NONE /print= MEDIUM.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 7 /method= HUNGAR /dlen= NONE /print= MEDIUM.

* PRINT= MEDIUM informs of diversity of weights in the co-occurrence matrix before reckoning the similarity. This information is the same from all the three runs of the macro. DLEN=NONE orders to output raw similarity as it is returned by method METHOD.
* GREEDY REORD tends to yield a similarity higher sometimes than GREEDY, albeit occasionally lower than HUNGAR. However, the differences usually do not exceed 1-2 percent.
* If you specify CHMAXW smaller (but not 1) – say, 2, the breakaway of GREEDY REORD from GREEDY will be expressed more (**Fig. 18**).



**Fig. 18**. Optimality for the example considered. Maximal possible reckonable similarity in matching, 100%, corresponds to the results of HUNGAR.

EXAMPLE 11. Maximal aligning method.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 7 /method= MXTRACE /dlen= NONE.

* This run uses aligning approach, not matching. The similarity size will be naturally lower than in EXAMPLE 10, because aligning does not try to maximize the number of co-occurrences accounted for when summing them up into the similarity.

EXAMPLE 12. Maximal common chain method.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= NOLIM /method= MXCHW /dlen= NONE.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 3 /method= MXCHW /dlen= NONE.

* These two runs are equivalent. **Tab. 1** in section “Algorithm” clarifies why. Suppose that the longest common chain existing in some pair of documents X and Y has length 6. Then CHMAXW=NOLIM will highlight that chain by weight 6. Being the sum of weights in the longest subsequence, the raw similarity will, consequently, equal 6∙6=62. From its own part, CHMAXW=3 will highlight that same chain by weight 3 (the limit of reward which the user permits). Therefore, the raw similarity will equal 6∙3. These results are equivalent, because we know how to make them identical: to take the root out of the raw similarity in the first instance, and to divide the raw similarity by 3 in the second instance.
* And really, if you request normalizing by DIVISOR1: replace /DLEN=NONE with /DIVISOR=DIVISOR1, the normalized similarity will come out the same in both cases. At normalizing, division by the parameter DIVISOR takes place. In the first case, that parameter (DIVISOR1 is *wmax*, see the normalizing formula in section “Algorithm”) in the pair X and Y will equal 6, and in the second case – equal 3.
* If not utilize s/c DIAG/TRV/TRSM, then METHOD=MXCHW is the method of longest common chain.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= NOLIM /diag= MAIN /method= MXCHW /dlen= NONE.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 3 /diag= MAIN /method= MXCHW /dlen= NONE.

* These two runs differ from the two previous by the use of DIAG s/c. Subcommands DIAG, TRV, TRSM are the subcommands which alter co-occurrence weights – relative those integer weights by which common chains are highlighted on the substage of chain rewarding. As a result of the altering, we have this:
* (i) if there exist several chains of maximal length, they may differ in weights: the most weighty one of them will win (will by itself represent the similarity); (ii) with CHMAXW=*number*, shorter chains than the longest one(s) may count “long” chains as well (see **Tab. 1**), so one of those shorter chains may win – if it turns out to be the most weighty out of the “long” ones.
* That’s why we speak, in general case – that is, in the prospect of DIAG/TRV/TRSM s/c – of “maximal common chain” method rather than of “longest common chain” method.

**DLEN**

This subcommand does optional normalizing of the raw similarity *S*(D1, D2), obtained by METHOD subcommand, into normalized similarity *Sim*(D1, D2), the similarity coefficient. The option concerns METHOD= GREEDY, HUNGAR, MXTRACE, and MXCHW, the methods where the raw similarity arises from summing up “useful co-occurrences” and a word participates in the summation no more than once. For other methods – MEANPW, MXW, SUM – the macro returns only raw similarity *S*(D1, D2) and so compels to specify DLEN=NONE.

DLEN is the normalizing length of the document; it enters the denominator in the formula of similarity coefficient *Sim*(D1, D2) between two documents (see “Algorithm: Normalizing into similarity coefficient”). The coefficient can theoretically attain the upper bound 1 only if DLEN=MIN or the two documents are of equal length. A document length |D| is the number of words in it (gaps are not counted). Parameter DLEN is chosen by the user based on their aims. You may also leave the similarity raw, unnormalized. Indicate one of the following:

MIN - (default/unspecifying) the length of the shorter of the two documents: min(|D1|, |D2|)

MAX - the length of the longer of the two documents: max(|D1|, |D2|)

MEAN - arithmetic mean length of the two documents: (|D1|+|D2|)/2

GMEAN - geometric mean length of the two documents: sqrt(|D1|∙|D2|)

HMEAN - harmonic mean length of the two documents: 2∙|D1|∙|D2|/(|D1|+|D2|)

NONE - do not normalize, return the numerator (raw similarity) *S*(D1, D2) as is.

With DLEN=NONE, arbitrary value 999 will stand on the diagonal of the square similarity matrix, otherwise it will be 1.

If you want to save document lengths, |D|, as a file, use s/c SAVEMXAW (see) under COMPARE=ALLPAIRS.

**DIVISOR**

This subcommand is allowed only if DLEN is not NONE. And it is ignored under CHMAXW unspecified or CHMAXW=1, i.e., when chain highlighting is not requested. Divisor *DIVISOR* enters the formula of normalizing the raw similarity into the coefficient of similarity. And its task is to remove from the numerator (the raw similarity) weights inflation associated with the reward of chains, so that the upper bound of the similarity coefficient would not exceed 1. See details in “Algorithm: Normalizing into similarity coefficient: Choice of value of DIVISOR”.

DIVISOR1 - “decision forgetting the weighting of chains”.

DIVISOR2 - “rigorous decision affirming the weighting of chains”.

DIVISOR3 - (default/unspecifying) “fair decision affirming the weighting of chains”.

DIVISOR4 - “compromise decision affirming the weighting of chains”.

If there are just two documents in the analysis (and s/c WMAX is unspecified), DIVISOR1 = DIVISOR2 = DIVISOR3.

**WMAX**

This optional subcommand, additional to s/c DIVISOR, allows you to specify your value for *Wmax* in place of the one determined by the macro. Use, if you want *Wmax* to not depend on the data. *Wmax* is utilized in variants DIVISOR= DIVISOR2, DIVISOR3, and DIVISOR4. See details in “Algorithm: Normalizing into similarity coefficient: Choice of value of DIVISOR”. Specify positive integer number.

**SAVEM1**, **SAVEM2**

These optional subcommands act if there are exactly two input documents. The subcommands save the co-occurrence matrix, in which rows are the sequence of elements of the 1st document (except the 1st row which displays the words of the 2nd document), and which columns are the sequence of elements of the 2nd document (and the 1st column, *SEQ1*, displays the words of the 1st document). Values in the matrix are the co-occurrence weights.

So, the 1st column and the 1st row – are the sequences (documents) themselves. They are given in the coding of variable *WORD\_.#$*, i.e., consist of positive integer codes. Negative values (these are random negative numbers) correspond to gaps between words, if there were any.

SAVEM1 will save the initial (obtained in stage 1 of the algorithm) co-occurrence matrix. SAVEM2 will save the co-occurrence matrix in its state entering stage 3 (see “Algorithm: Reckoning raw similarity”, see also **Fig. 16**).

Specify one or both subcommands. In each, indicate own external .SAV file (path/name in quotes or apostrophes) or a name of a declared dataset (optionally in quotes or apostrophes), for saving[[8]](#footnote-8).

S/c SAVEM2 is incompatible with COMPARE=FIRSTEXT.

EXAMPLE 13.

dataset declare mat2.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= NOLIM /savem2= mat2 /method= GREEDY.

* The co-occurrence matrix on the basis of which the reckoning of similarity takes place is saved as a new dataset *MAT2*.

**SAVEMXAW**

This optional subcommand (“save maximal awarded weight”) saves, for every comparison done, the greatest weight (value) in the co-occurrence matrix, observed after the matrix creation and the optional highlight of chains – that is, after stage 1 or after substage 2a, if there it was (see “Algorithm”). Specifically:

* If CHMAXW=1 (= by default of CHMAXW), i.e., there was no rewarding of chains, the maximal weight being saved is the weight from the initial co-occurrence matrix. (It may be binary or fractional, depending on whether s/c TRV/TRSM was specified.)
* If CHMAXW is not 1, i.e., there was rewarding of chains, the maximal weight being saved is the weight from the co-occurrence matrix after the reward of chains, i.e., after the actions of CHMAXW and CHMINW subcommands. It is an integer weight. And, if you have s/c TRV/TRSM specified, the weight being saved is the one before multiplying it by the initial, fractional weight (see “Algorithm: Weighting by chain length in situation of TRV or TRSM subcommands; see also **Fig. 16**). In the “Algorithm”, it is denoted *wmax*.

Indicate an external .SAV file (path/name in quotes or apostrophes) or a name of a declared dataset (optionally in quotes or apostrophes), for saving. If COMPARE=ALLPAIRS, square symmetric matrix will be saved, with the document lengths (not counting gaps) standing on its diagonal. If COMPARE=FIRST, just vector is saved.

S/c SAVEMXAW is incompatible with COMPARE=FIRSTEXT.

EXAMPLE 14.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= NOLIM /savemxaw= 'd:\exercise\mxw.sav' /method= GREEDY.

!KO\_seqsim docvar= seq /wordvar= word /chmaxw= 5 /diag= MAIN /dlen= MEAN

/savemxaw= 'd:\exercise\mxw2.sav' /method= GREEDY.

* In the 1st, preliminary run, CHMAXW=NOLIM, so the matrix being saved by SAVEMXAW will contain the length of the longest existing concurrence chain. Having opened and inspected the file *MXW.SAV*, the researcher found that these values in his data lie around 5-7. Therefore he decided to set in the 2nd, final run, CHMAXW on 5. Common chains of length 5+ will be awarded maximal weight 5.
* In the 2nd run, the researcher entered some options that he planned. By the way, values in *MXW2.SAV* will no more be above 5.

**PRINT**

Optional PRINT prints out in the output Viewer some information, including report on the initial autorecoding.

SHORT - raw similarity *S*(D1, D2) computed, also a note if the matrix became zero (empty) and when. (The matrix might have been empty initially, or become empty in the course of chain highlighting.)

MEDIUM - above that, information on density (% of the number of cells) and diversity on nonzero values, i.e., weights, in the co-occurrence matrix before reckoning the similarity. If RESCRND is specified, the information is displayed twice, before the rounding and after it before the similarity reckoning, – what might be of use in order to tune the macro (see **Fig. 16**).

LONG - above that, the co-occurrence matrix itself (if RESCRND specified, then twice). When DIAG or TRV/TRSM are specified, that is, values in the matrix could be fractional, they are displayed with 4 decimal digits. There is no sense to use LONG if documents are big: printout is uninformative then and, besides, slows down the macro’s performance.

**DATASET**

This subcommand is only in effect and is required with COMPARE=FIRSTEXT or if TRSM is specified, and it is of an auxiliary technical character. Indicate the name of the input, working dataset. Or specify DATASET=RENAME, then the macro will simply rename your input dataset into *KO\_SEQSIM#$.\_*.

**NEWDATA**

Optional command to set a name for the output dataset with the similarities (or with the longest chains, if COMPARE=FIRSTEXT). Specify a name. By default/unspecifying, the dataset will remain untitled.

***Special regimes***

The macro ignores weighting and is not suited for split state of the dataset. It turns off weighting of the dataset. The macro obeys commands selecting cases (FILTER, USE, SELECT IF, N OF CASES). It does not obey temporary (standing under TEMPORARY) operations.

***Appendix***

Data for **Fig. 17**.

\*Fig. 17a data.

data list free /word (a8).

begin data

A B C D E F G H I J K L M N O L P Q R S T T U V W X T Y Z AA BB CC

DD EE C GG D K J I O K J S HH L II JJ T O L R T KK W KK T MM X X V Z Y NN OO BB PP

end data.

string seq (a8).

do if $casenum<=32.

compute seq= 'seq1'.

else.

compute seq= 'seq2'.

end if.

execute.

dataset name fig17a.

\*Fig. 17b data.

data list free /word (a8).

begin data

A B C D E F G H I J K L M N O P M Q R H S T I J K F U V W X Y Z

AA BB CC O P DD EE H I J K F FF GG R HH D E F U V II B JJ KK I J K L M Q LL H MM NN

end data.

string seq (a8).

do if $casenum<=32.

compute seq= 'seq1'.

else.

compute seq= 'seq2'.

end if.

execute.

dataset name fig17b.

\*Fig. 17c data.

data list free /word (a8).

begin data

A B C D E F G H E F I J K L M N O K P Q I R S T U V W X U Y Z AA

BB C D CC H DD D E F I J K EE J K L M I R S FF GG K P HH II JJ V W X U KK LL MM NN

end data.

string seq (a8).

do if $casenum<=32.

compute seq= 'seq1'.

else.

compute seq= 'seq2'.

end if.

execute.

dataset name fig17c.

1. Not to confuse with the *sequence optimal matching* founded on Needleman–Wunsch algorithm. [↑](#footnote-ref-1)
2. Only under subcommand TRSM a co-occurrence of a term not with itself can be paid attention. [↑](#footnote-ref-2)
3. The formula of function Φ is borrowed from Werner Antweiler (<https://wernerantweiler.ca/blog.php?item=2018-11-03>), who has named it sigmoid-logit distribution function. [↑](#footnote-ref-3)
4. or not this – in the condition of s/c TRSM with non zero similitudes. [↑](#footnote-ref-4)
5. It is the simplest method of sequence comparison by means of alignment; there are other methods of alignment, in particular, those inserting gaps here in there inside a sequence and doing other edits, but these methods are not offered by the present macro. [↑](#footnote-ref-5)
6. It is this weight *wmax* subcommand SAVEMXAW will save. [↑](#footnote-ref-6)
7. It is this weight *wmax* subcommand SAVEMXAW will save. [↑](#footnote-ref-7)
8. If you wish to plot an image of the matrix saved, a heatmap, use macro !KO\_EDPROXMX (“Various proximities” collection). [↑](#footnote-ref-8)