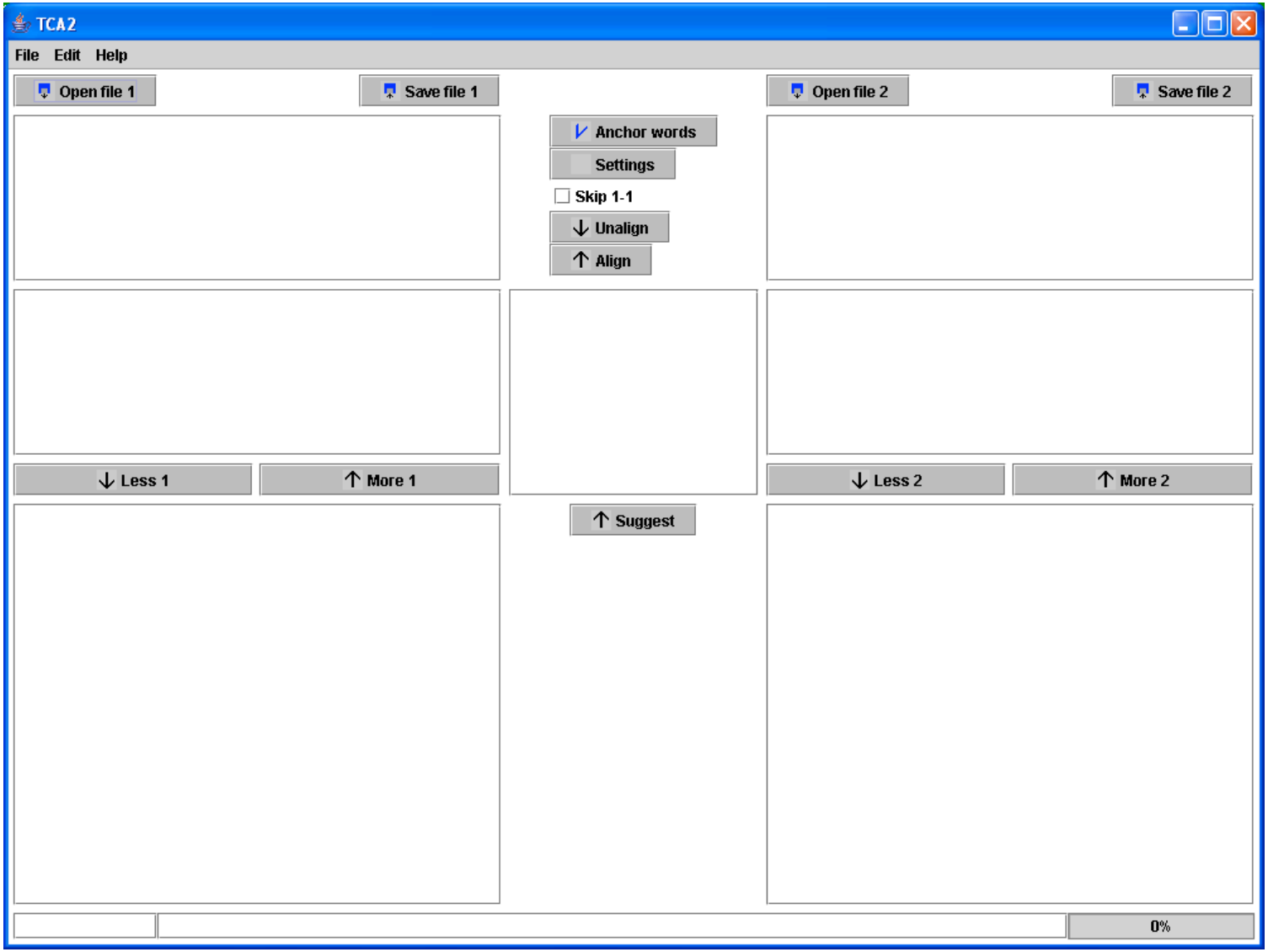
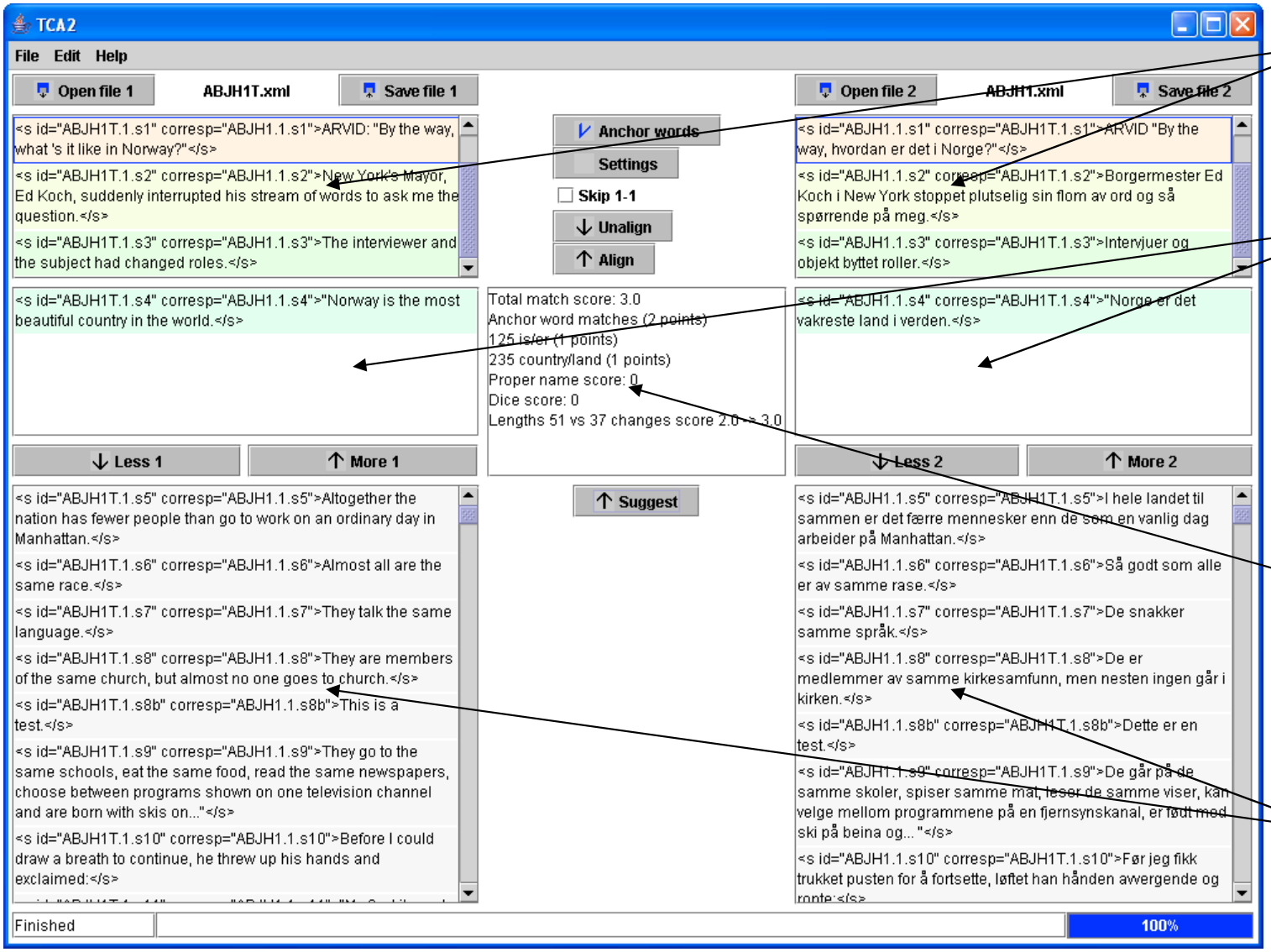
TCA2

Øystein Reigem (current developer)
Johan Poppe (earlier developer)
Knut Hofland (project leader)
Gjert Kristoffersen (project leader)

Aksis, University of Bergen

2005


TCA2
File
Edit
Help
Open file 1
Save file 1
Open file 2
Save file 2
Anchor words
Settings
Skip 1-1
Unalign
Align
Less 1
More 1
Less 2
More 2
Suggest
0%

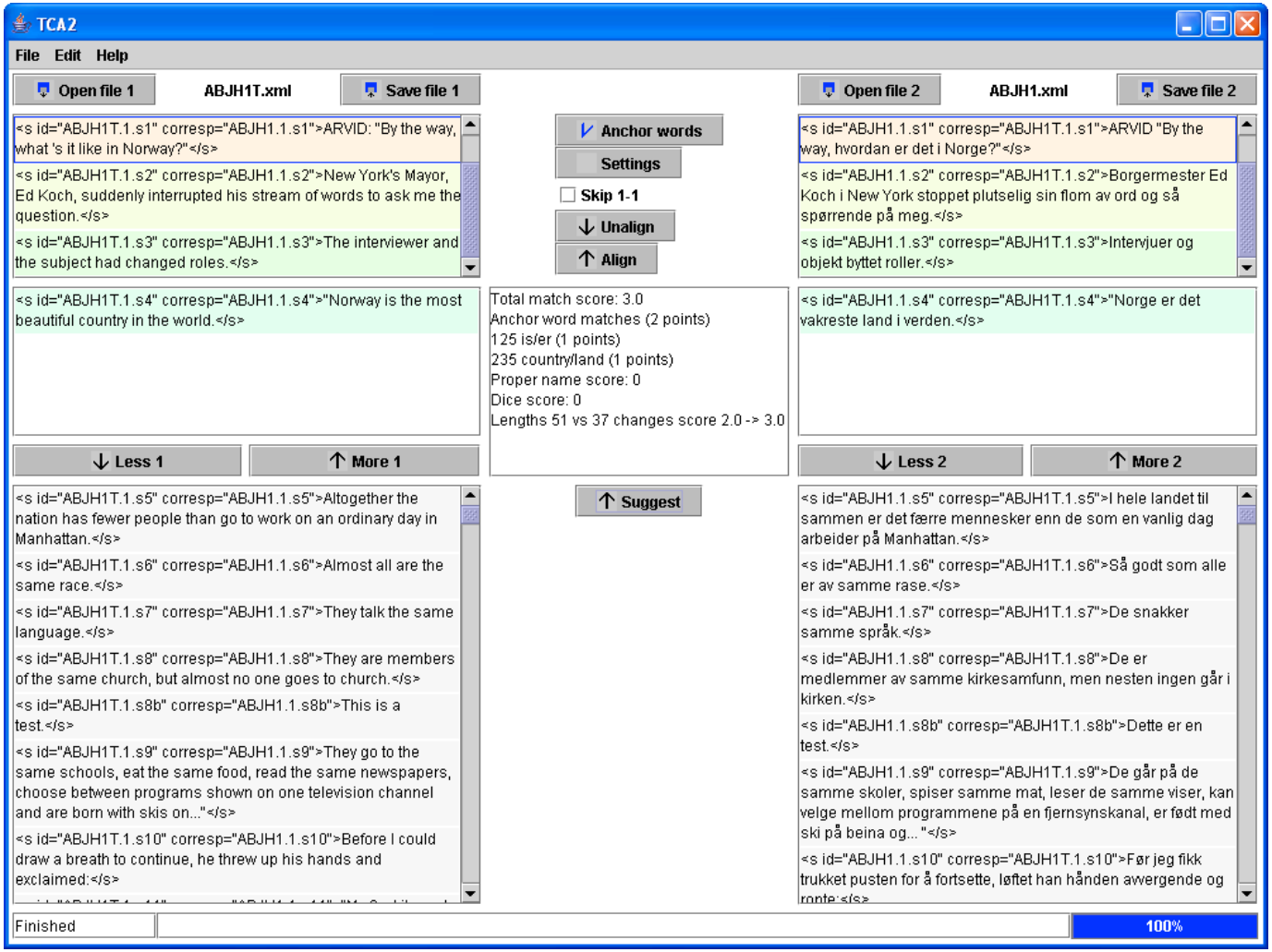


Area for aligned elements

Area for elements to align, i.e, elements awaiting approval or changes

Box that shows how well the elements to align match

Area for unaligned elements

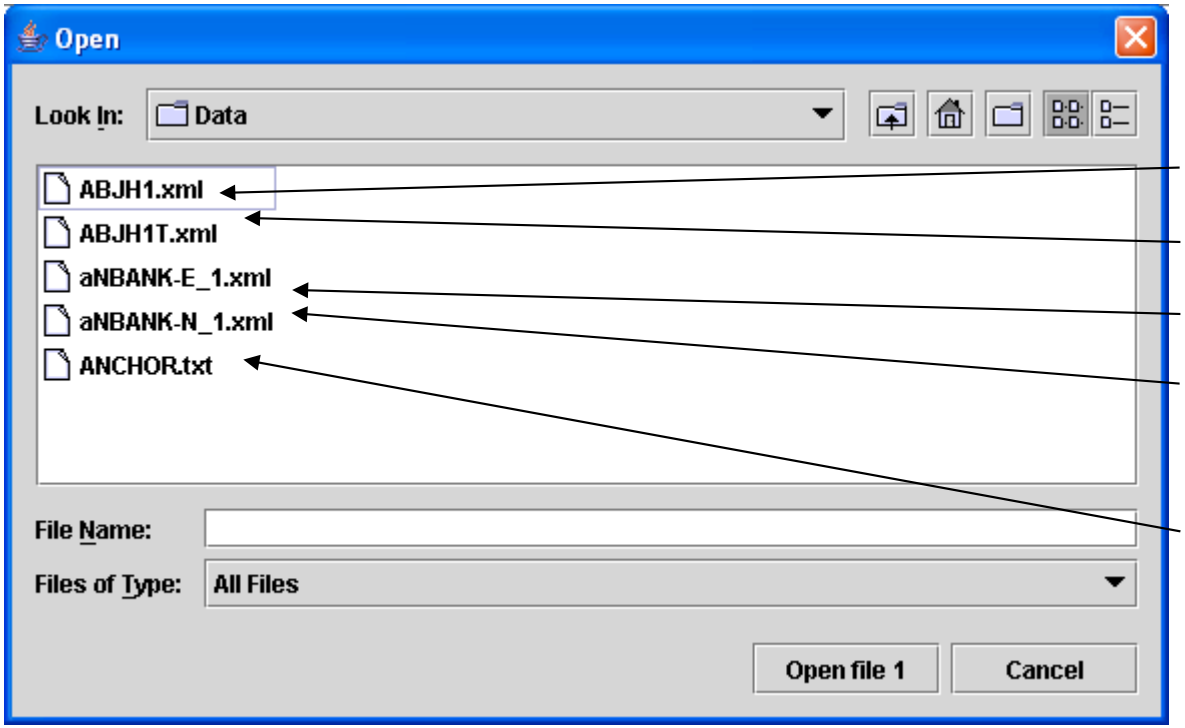


The application shows elements exactly as they appear in the *input* file, tags and attributes and all.

The application uses 'corresp' attributes to represent alignments in the *output* files. Existing 'corresp' attributes - if any - will be stripped. But even if the app changes the coding, the changes are not shown in the interface. The 'corresp' attributes we see in this illustration are old ones present in the input files. (The files used for these illustrations are files that have already been aligned with a different program.)

Opening files - texts and anchor word list.

These are the files of the demo system.



Norwegian

English

English

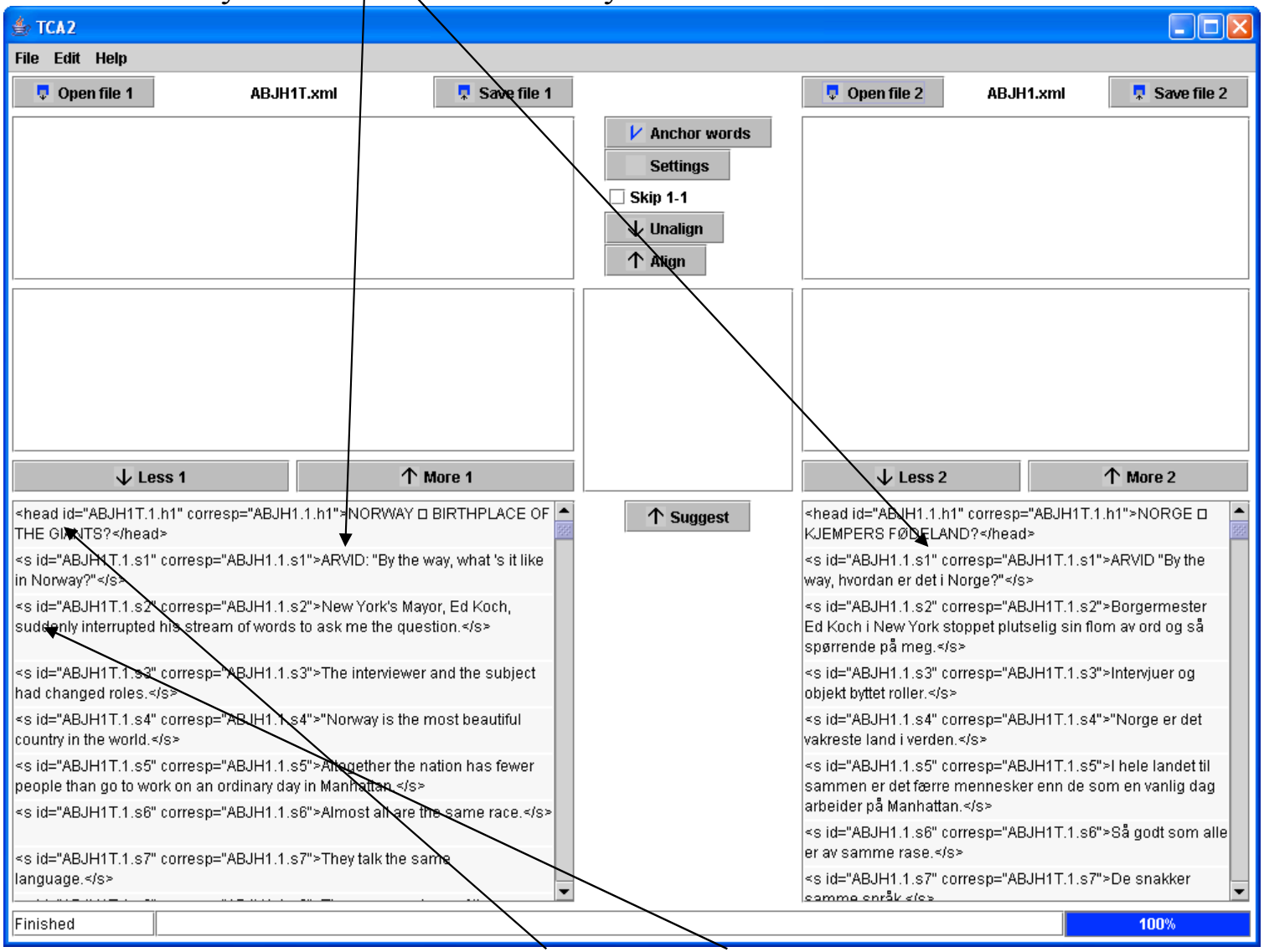
Norwegian

English/Norwegian anchor word file

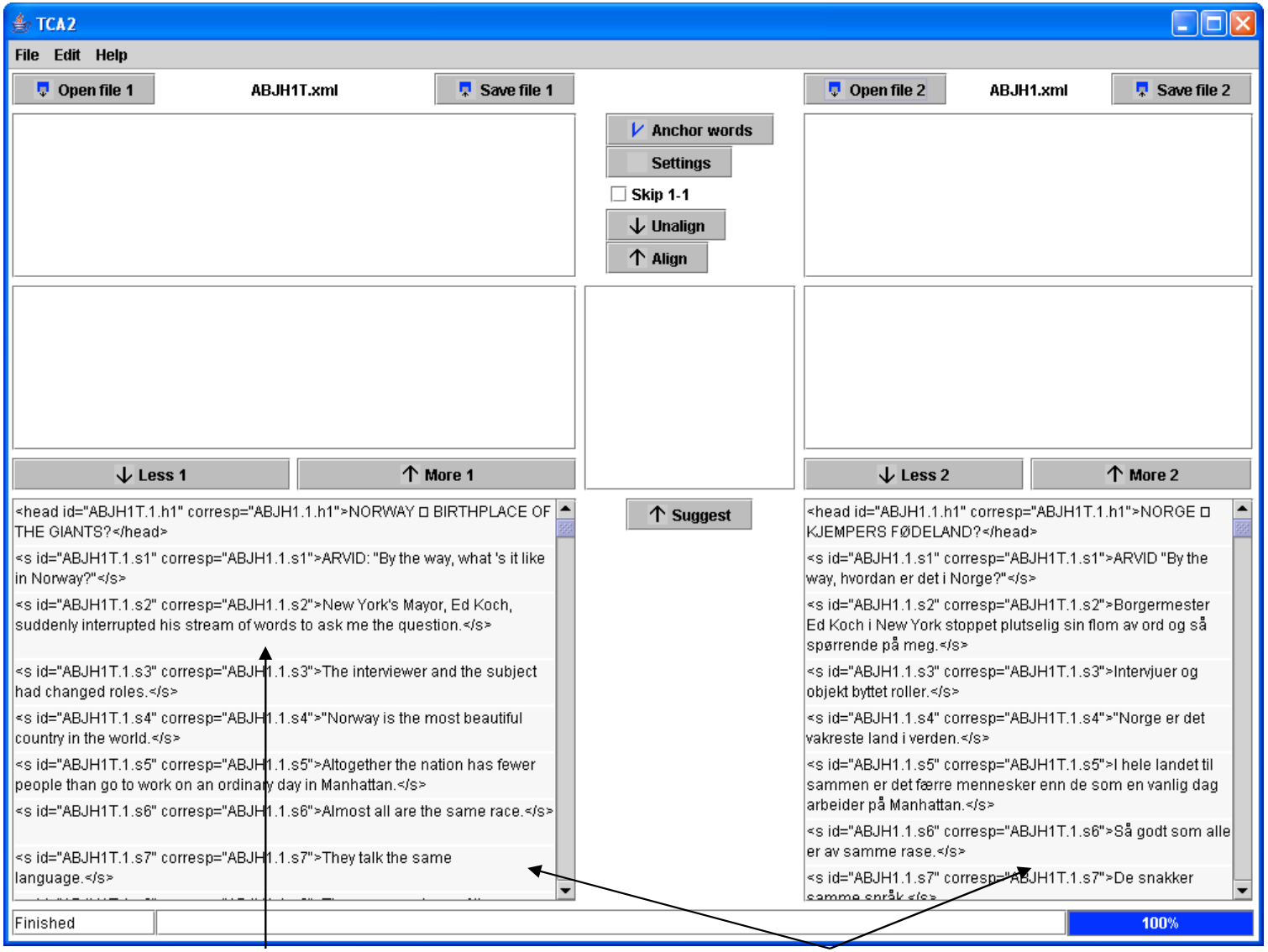
Texts.
Must be xml.
Must contain elements that the app recognizes.
The elements must have 'id' attributes with unique values

Open English text on left side, Norwegian on right

Bug: Sometimes a text doesn't show after the file have been read in.
Possible remedy: **More** button followed by **Less** button.



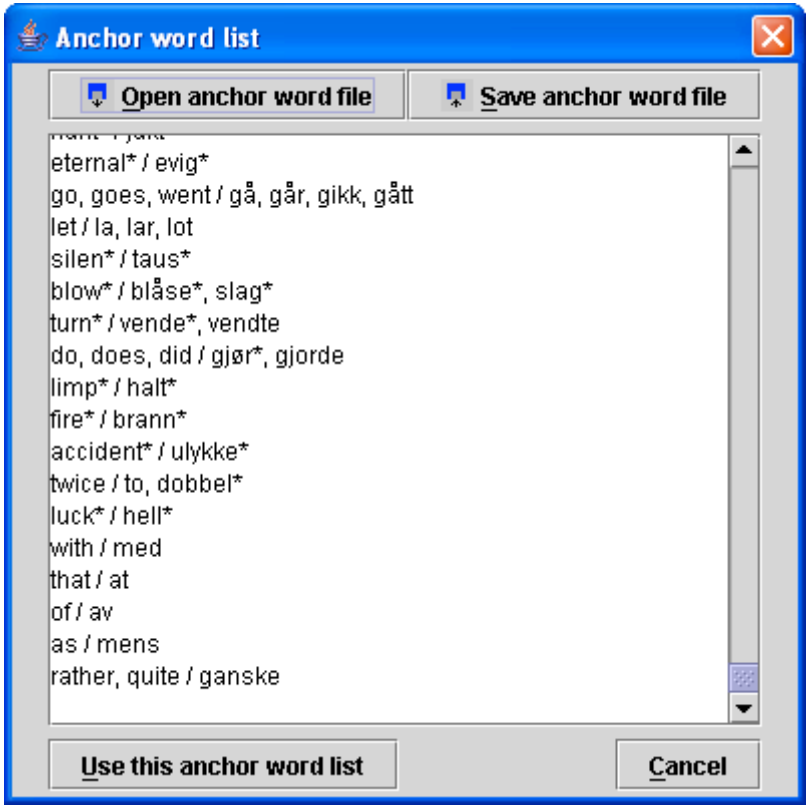
Application currently set up to handle <head> and <s> elements. Later: Settable from **Settings** button.



Bug: Size (height) of cells doesn't always fit the content. Cells are sometimes too large, sometimes too small. Resizing the window might make matters worse.

Bug: Unequal widths.
Tip: Start app. Open both files. Then maximise (or increase width).

Bug: Status from reading in files doesn't go away.



Wildcard '*' can be used anywhere, e.g, *ulykke*

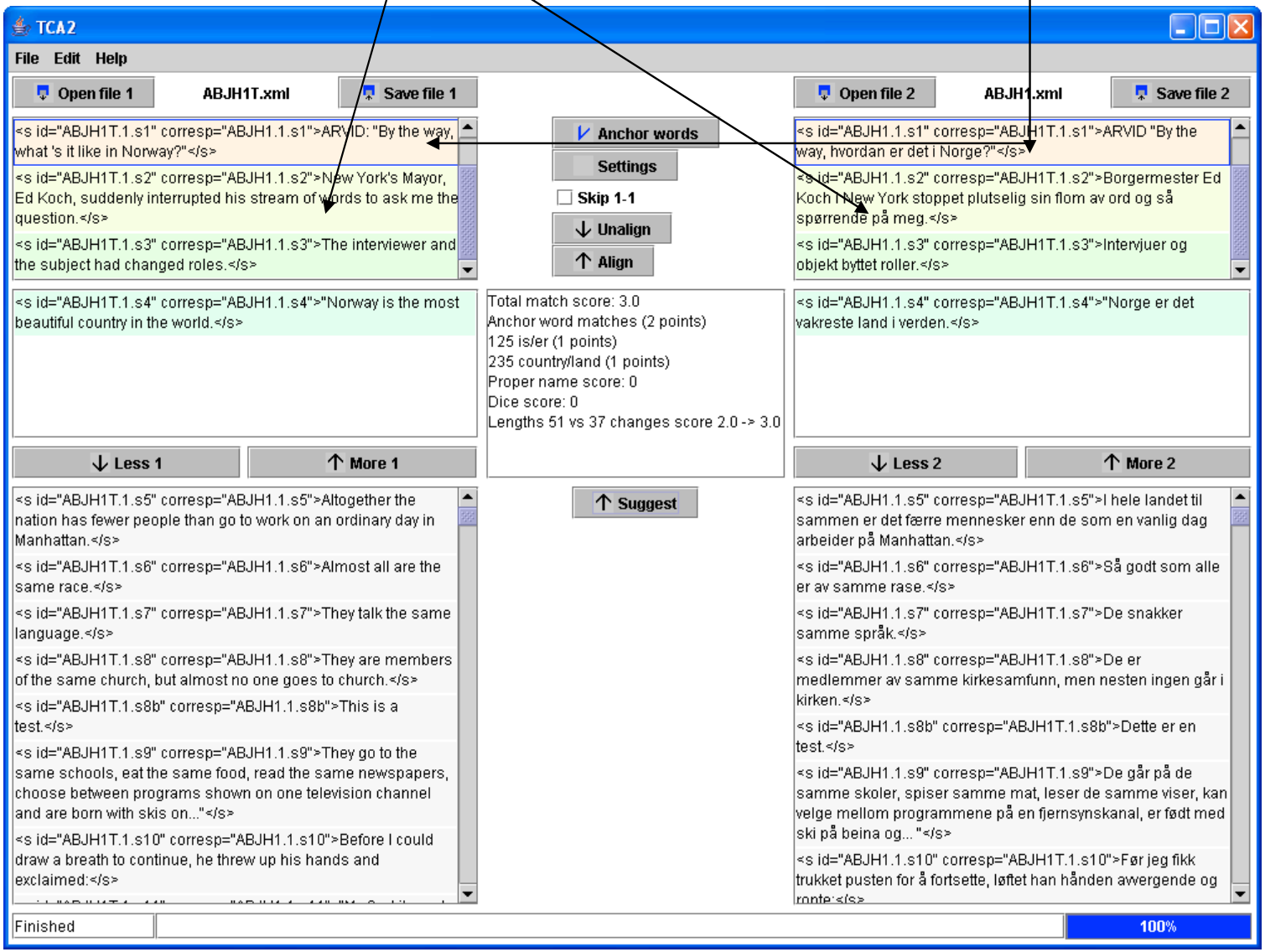
Phrases possible, e.g, red wine,claret/rødvin

List can be edited and saved back to file.

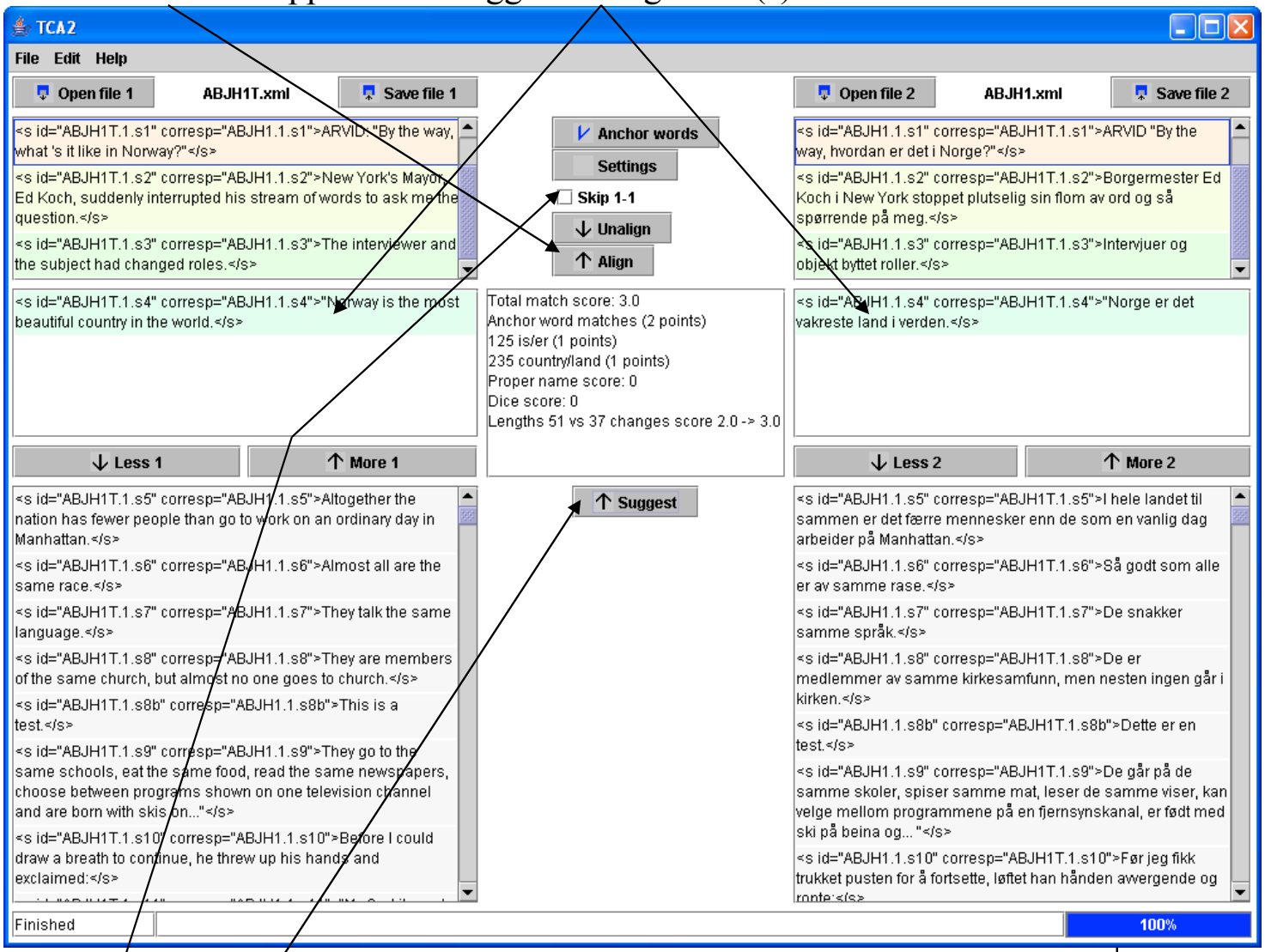
Anchor word list handling is buggy. Open list before alignment starts. Do not attempt to re-open file or do any changes.

Colours are used to show which elements are aligned

Clicking on an element in one box will scroll the other box to get alignments in synch visually

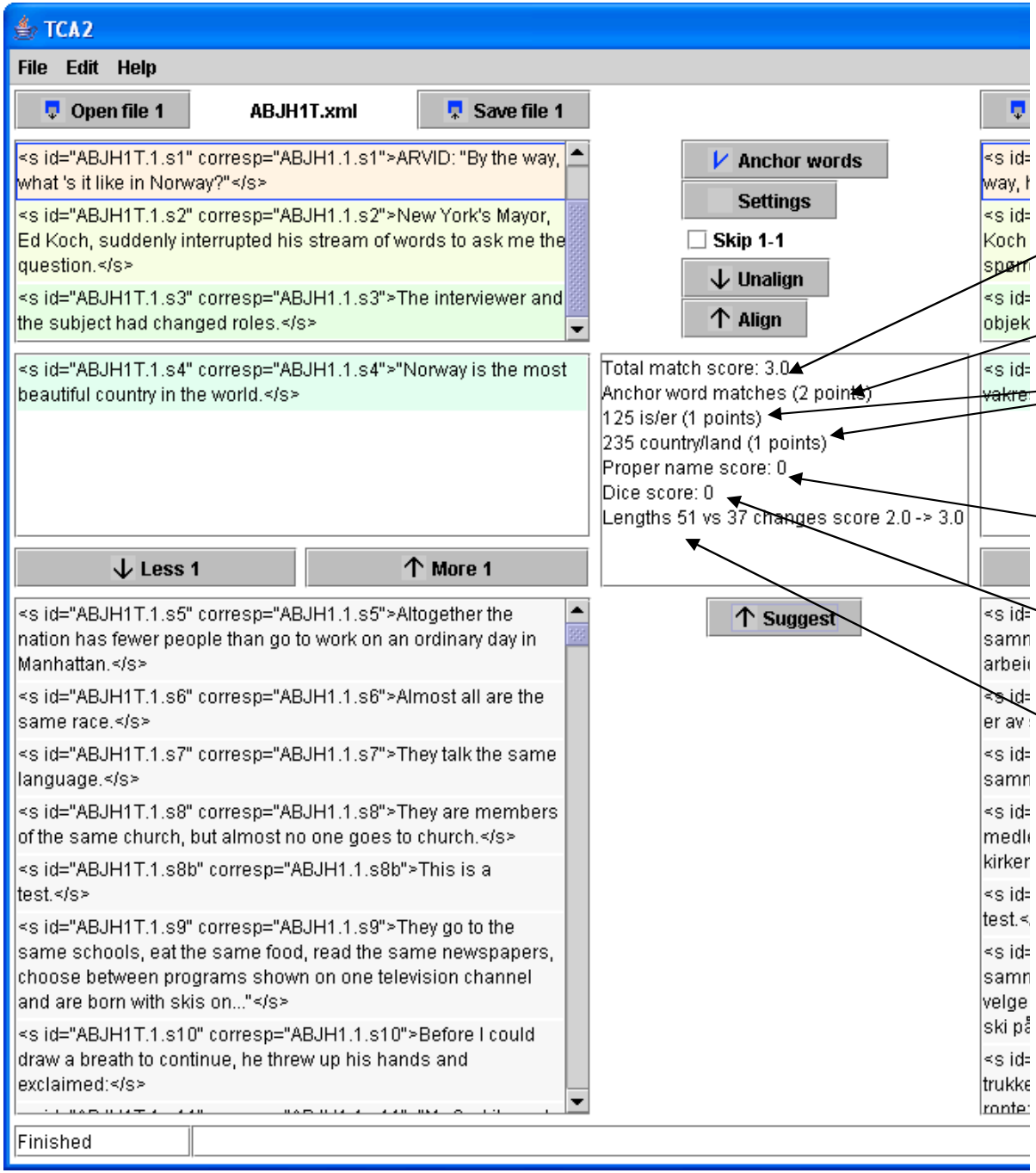


Use this button to approve the suggested alignment(s)



Press the **Suggest** button to get the application to suggest a new alignment. If the **Skip 1-1** box is ticked the app will automatically approve its own suggestions as long as they are 1-1 alignments, and not pause for approval until some other kind of alignment appears (0-1, 1-0, 1-2, or 2-1).

Note: Not possible to press **Suggest** button while there still are pending alignments. App needs '**Align&Suggest**-in-one' feature.



Total score, based on

(1) Anchor word match

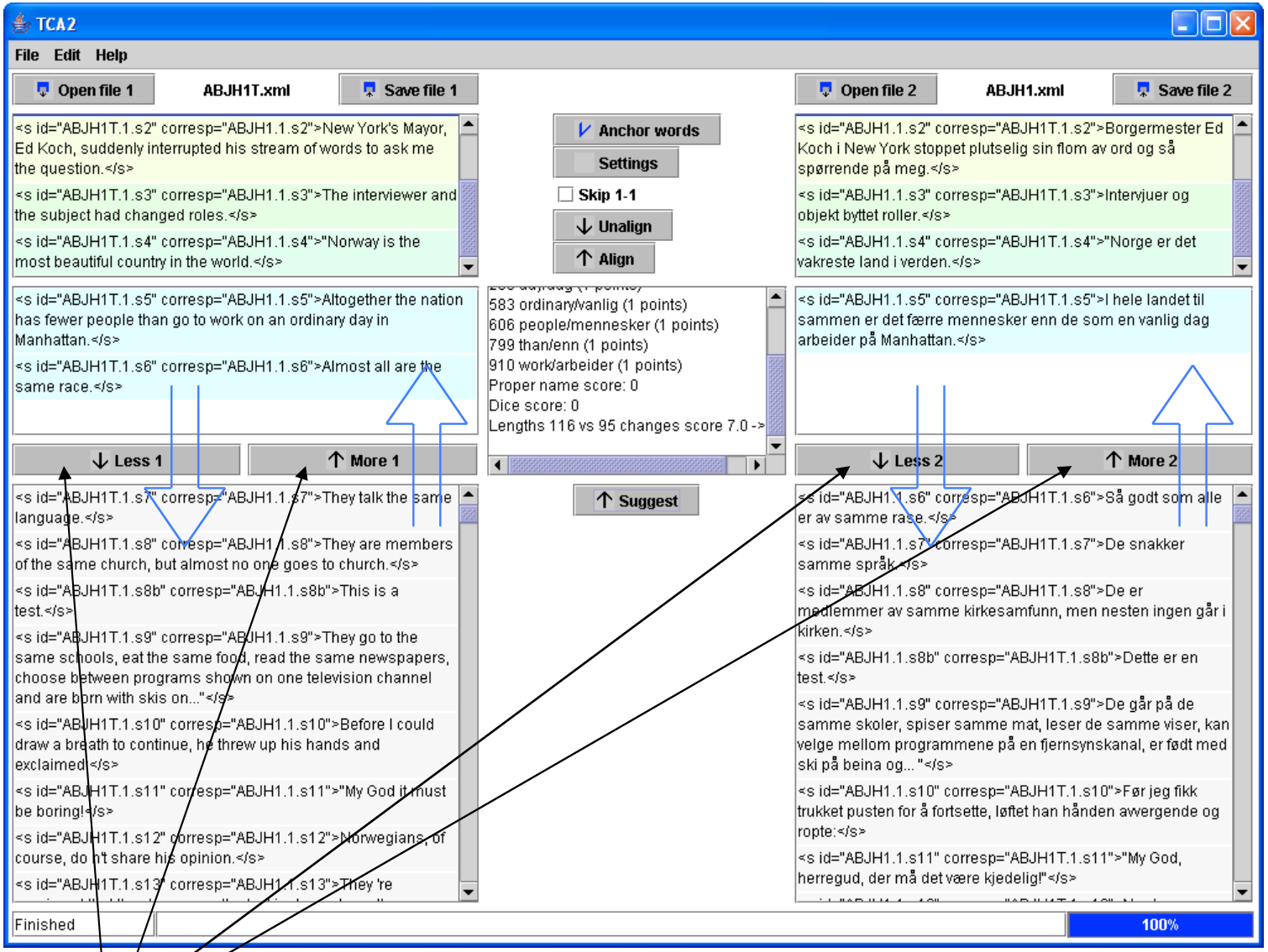
(Here are details about anchor word matches)

(2) Proper names match (not implemented)

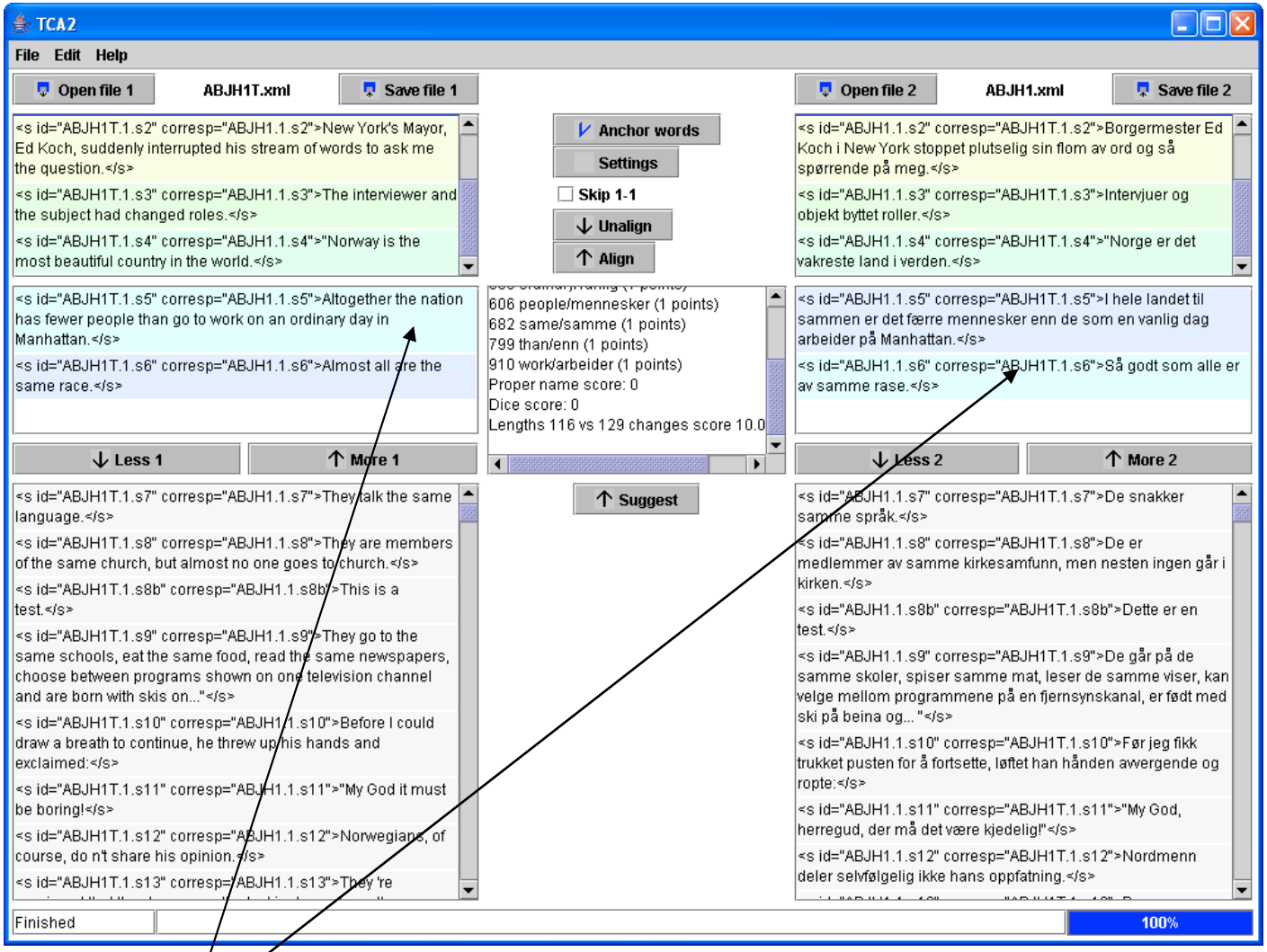
(3) Dice comparison (not implemented)

(4) Length match

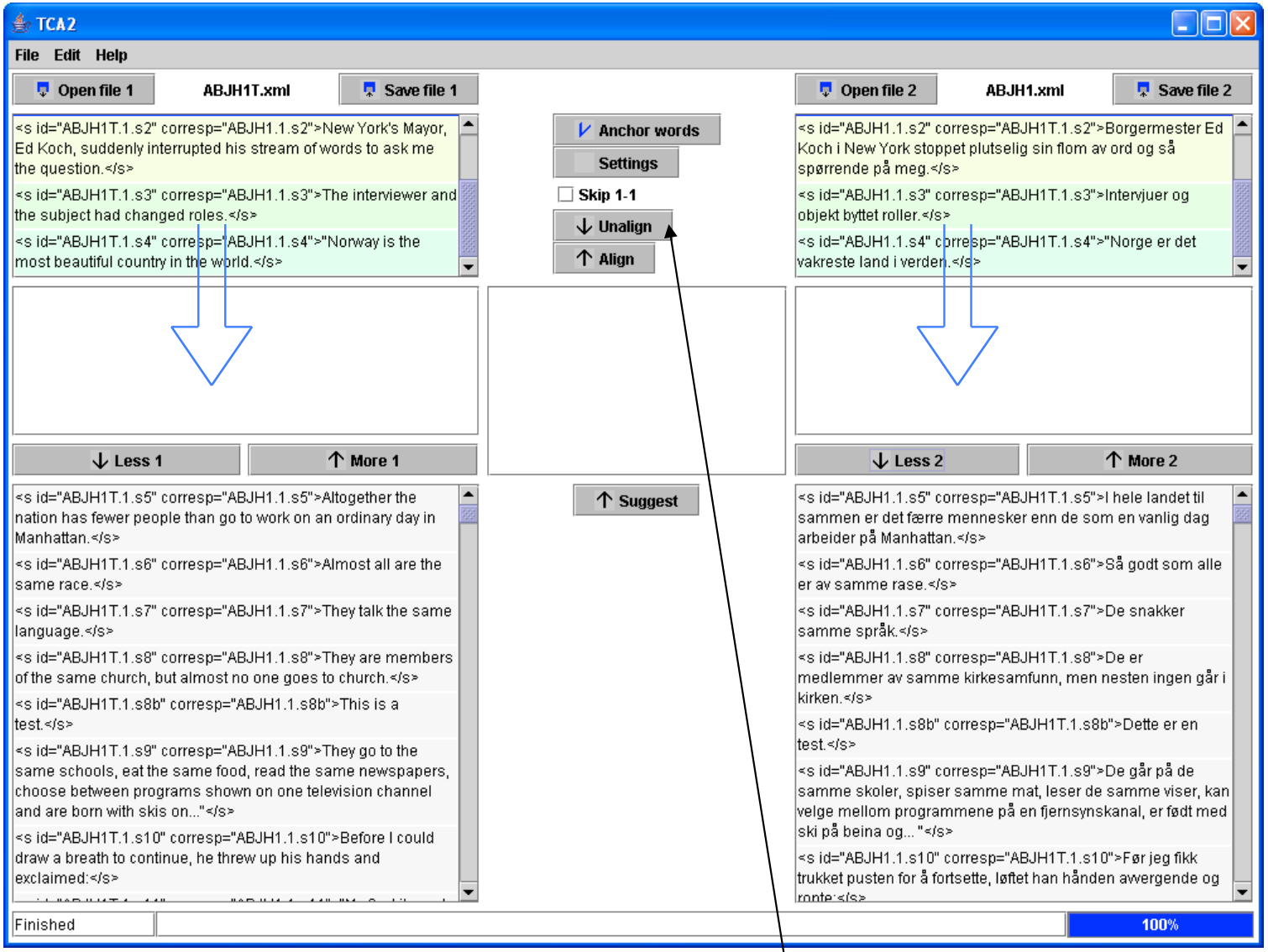
Scores from (1), (2) and (3) are added together, and modified according to (4)



Click these buttons to manually change the sets of pending elements.



Click elements to change the relations between them. Can be used to establish crossing alignments, like here. Not really needed for anything else. (Can not be done for already approved alignments.)
When clicking, relations (colours) sometimes change in unexpected ways (but there's a logic behind it).
Sometimes an element must be clicked more than once to achieve the needed relation.



The most recent alignment(s) can be undone with the **Unalign** button.

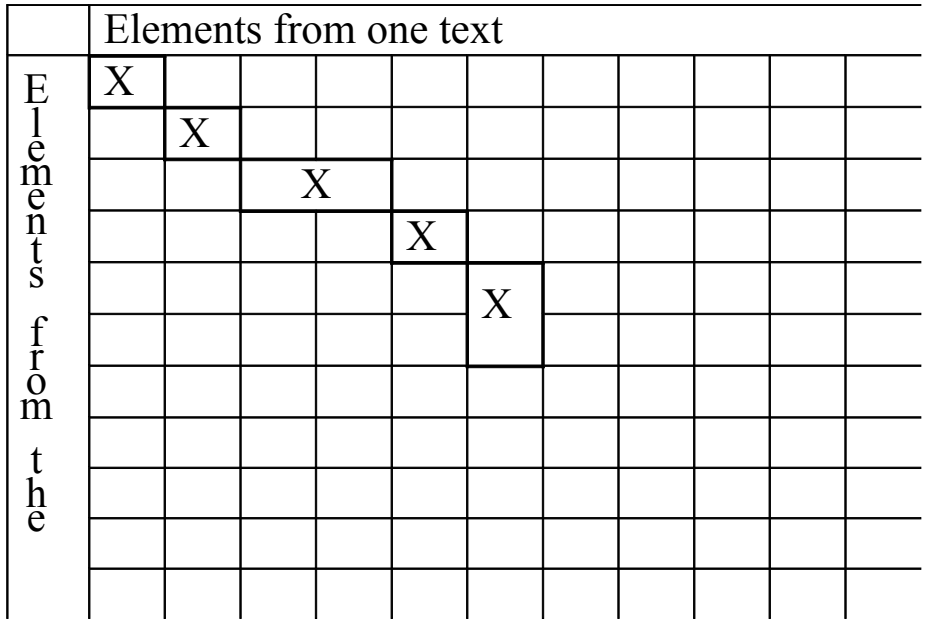
When the app suggests an alignment it first tries out various combinations of elements and alignments.

Each single alignment it tries takes some elements (say **n** elements) from the first text and combines with some elements (say **m** elements) from the second text. Allowable **m-n** are 0-1, 1-0, 1-1, 1-2, 2-1.

Each such combination gets a score according to several criteria, e.g, the occurrence of matching anchor words in the two sets of elements. (0-1 and 1-0 combination get a score of 0.)

The various possible combinations are not evaluated in isolation. Instead the app looks further in the texts and tries out whole paths of combinations.

Example: Here is one such path with combinations 1-1, 1-1, 2-1, 1-1, 1-2.



(This path has no 0-1 or 1-0 combinations, but they are so tricky to draw.) :-)

Each path gets a score which is the sum of the scores of the individual combinations.

The app tries out all such paths of a certain length (a certain number of steps) before deciding on the best one (the one with the highest score). During the process it prunes sub-optimal paths.

(Combinations 0-1 and 1-0 contribute with 0 to the path score, but paths containing such combinations may still sometimes get a high enough score from the other steps to win.)

After selecting the best path the app selects the first step of that path as its suggestion for an alignment. (That alignment might not be the winning one if only the first steps were compared.)

(So there is a lot of work behind arriving at just one single alignment. But the app doesn't throw away all its temporary results. Much of them will be reused when working on the next suggestion. Btw - garbage collection hasn't been implemented yet, and some temporary results pile up in memory.)

The application has a button and a dialog box for changing various settings. Currently the only setting in the dialog is the length of paths to try (with a rather low default value of 3). Among other possible settings in the futere might be the names of elements to handle (currently only <head> and <s>.)