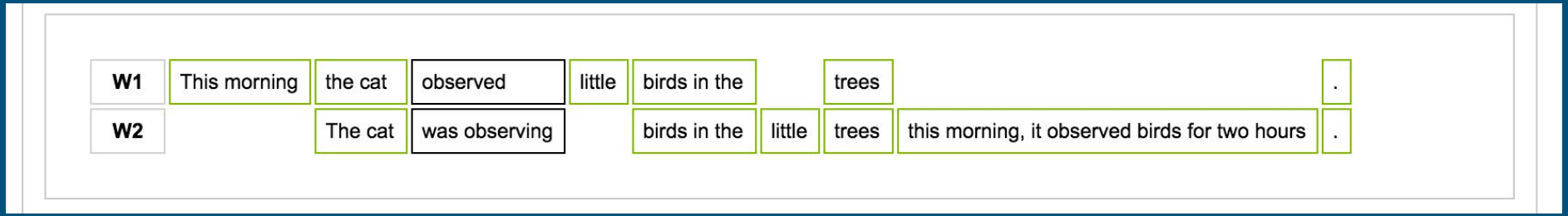
A tool for syntax-based intra-language text alignment

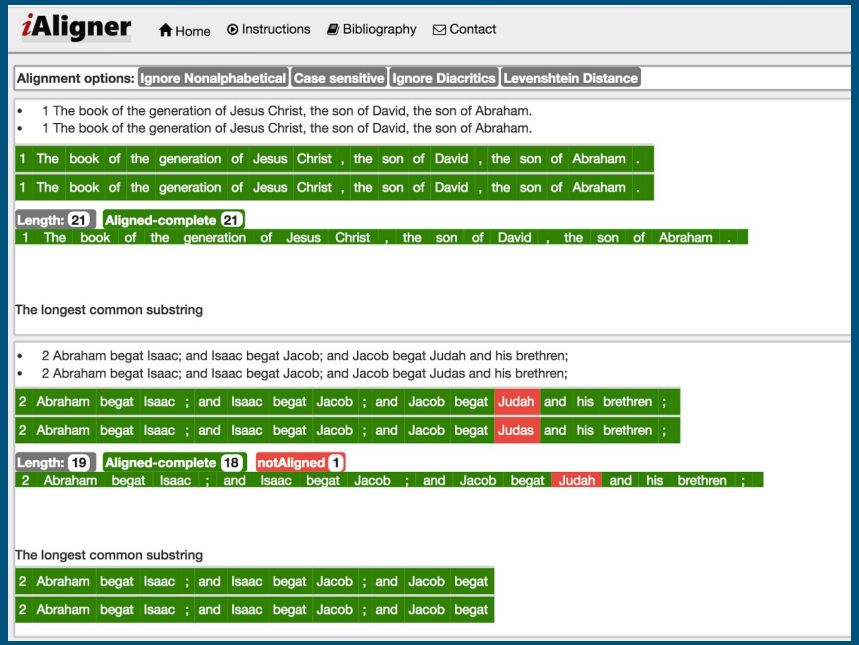
Tariq Yousef, Chiara Palladino
University of Leipzig
Berlin Digital Classicist Seminars,
November 29, 2016

What is text alignment?

- Text alignment is the comparison of two or more parallel texts
- It tries to define correspondences/similarities and divergences/variants
- One of the most important tasks in Natural Language Processing: it can be performed automatically through algorithmic and dynamic programming methods



Intra-Language alignment: alignment of texts in the same language



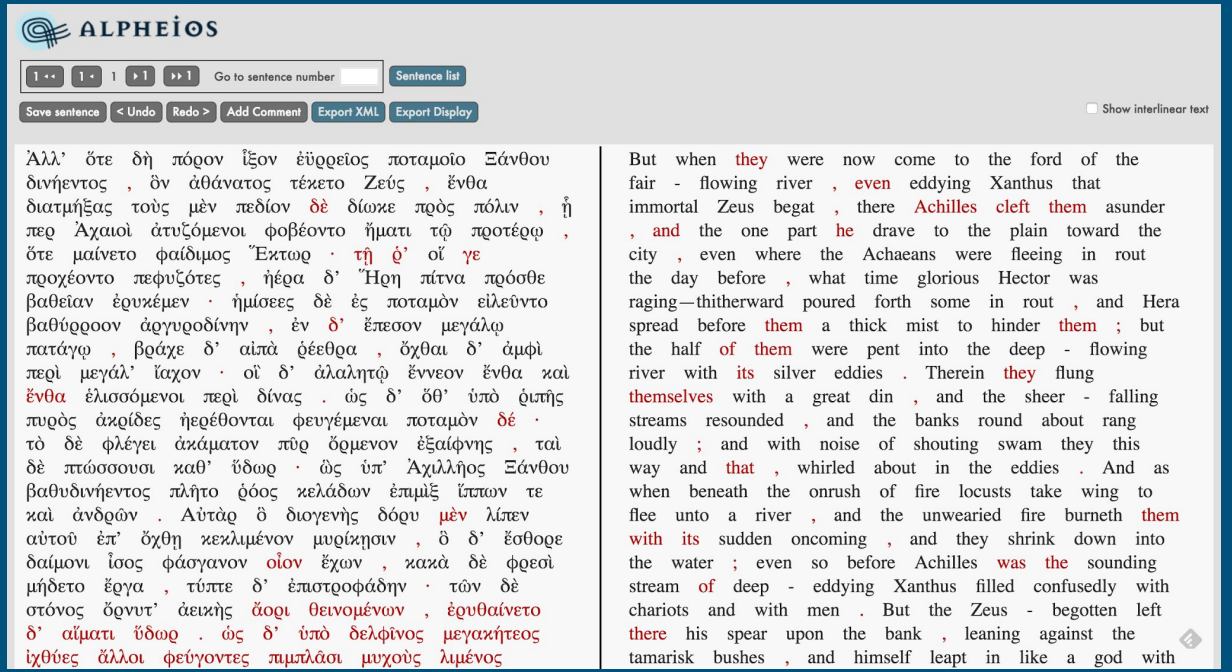
Cross-language alignment: alignment of texts in different languages

- Cross-language alignment is difficult to perform automatically
- It still needs training data from manual alignment

A Persian poem manually aligned with an English translation, from the project Open Persian (http://www.dh.uni-leipzig.de/wo/open-philology-project/open-persian/)

...So, there is also manual alignment

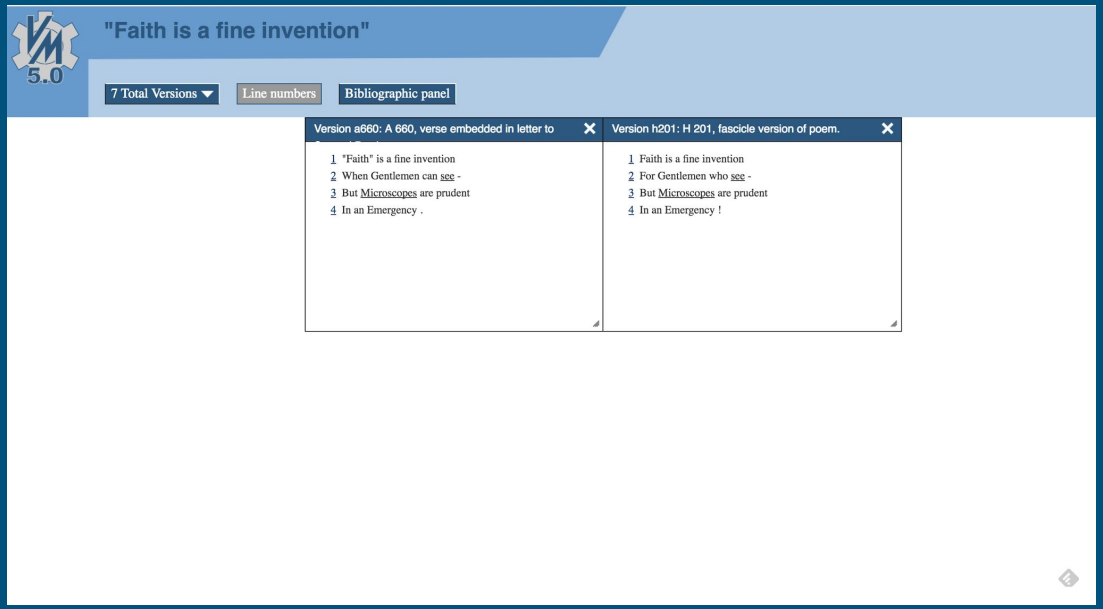
- The Perseids Project and Alpheios Texts provide tools for manual alignment of texts in different languages (http://www.perseids.org/, http://alpheios.net/)



Homer, *Iliad* XXI, aligned with the English translation by A.T. Murray

Pairwise alignment: alignment of two texts

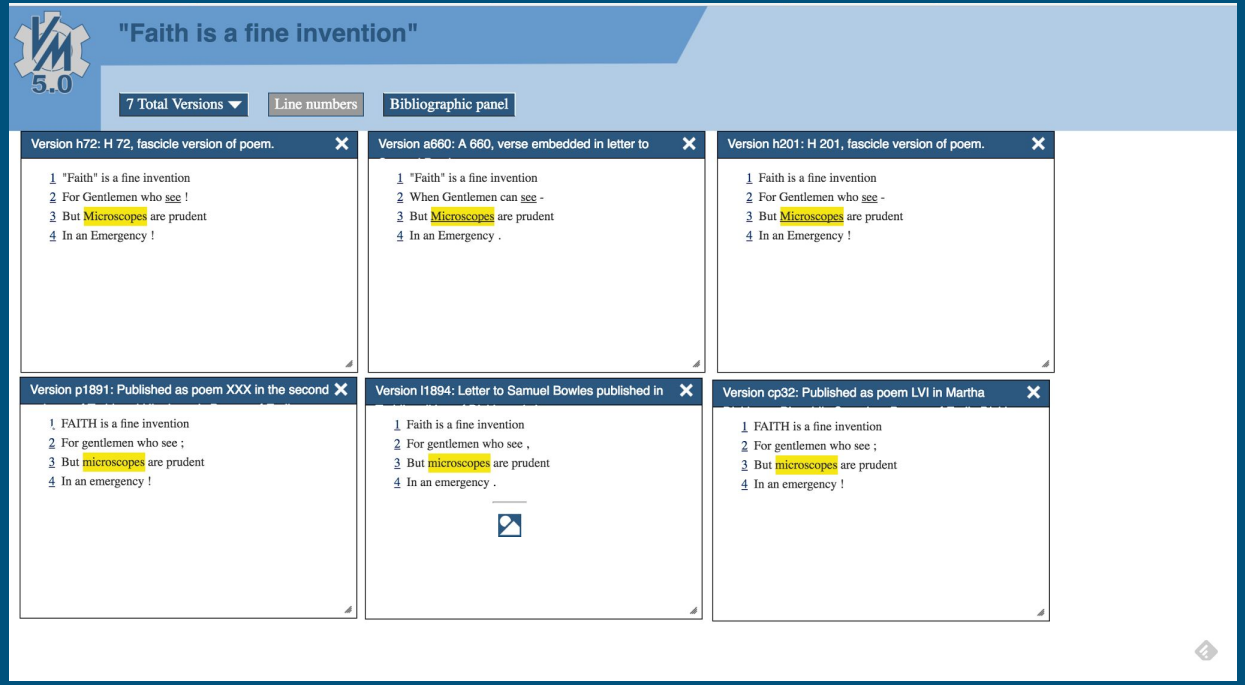
We distinguish on the number of text because it determines differences in the use of the alignment algorithm



Two versions of Emily Dickinson's *Faith is a fine invention*, aligned using the Versioning Machine (http://v-machine.org/samples/faith.html)

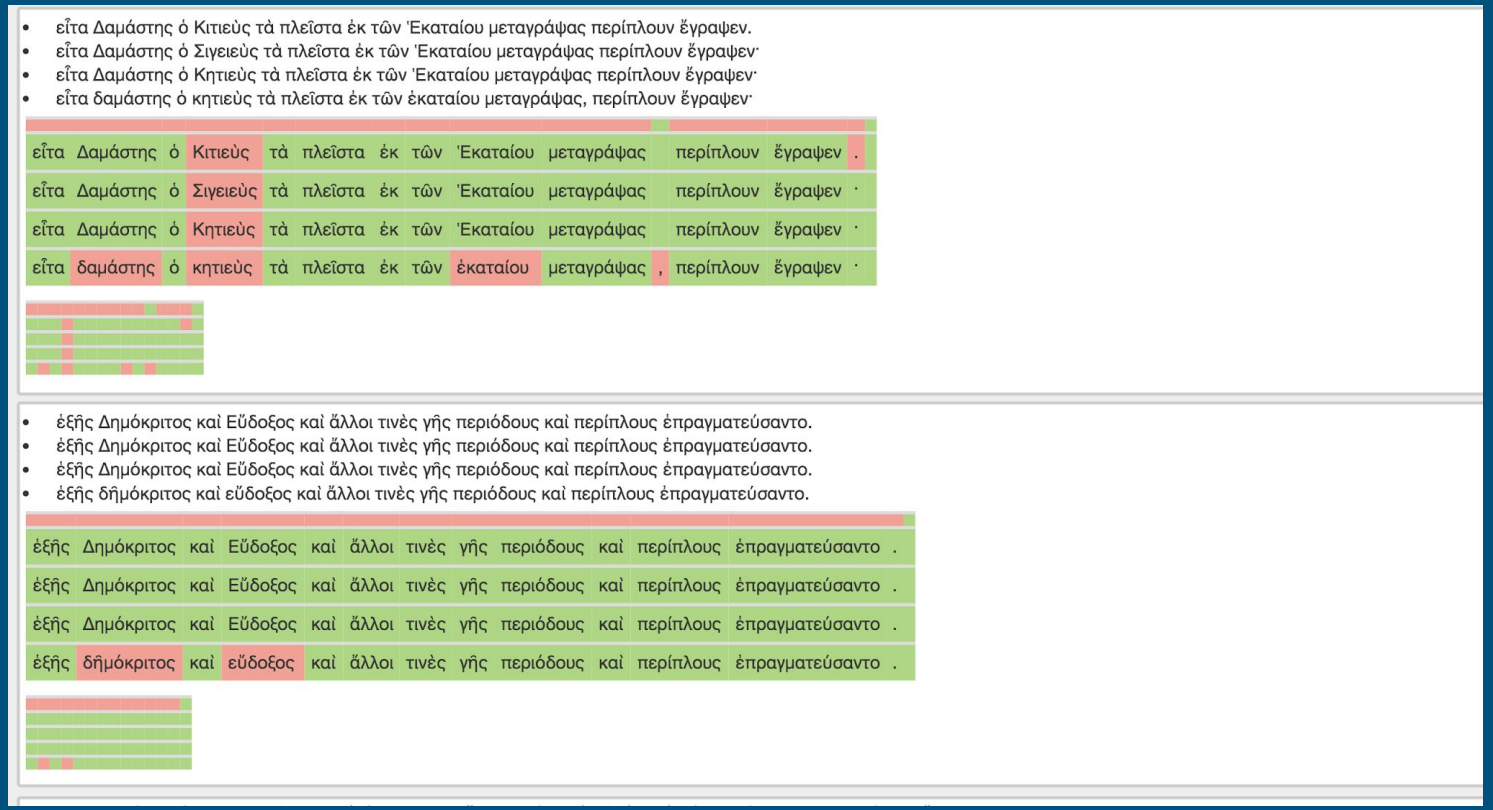
Multiple alignment: alignment of multiple texts (i.e. more than two)

The number of multiple texts is virtually unlimited: in an ideal world, you can align as many texts as you want (but you should be careful and avoid “alignment monsters”)



Six versions of the same poem by Emily Dickinson

Four texts aligned with iAligner



Alignment can be visualized in different ways

As a table

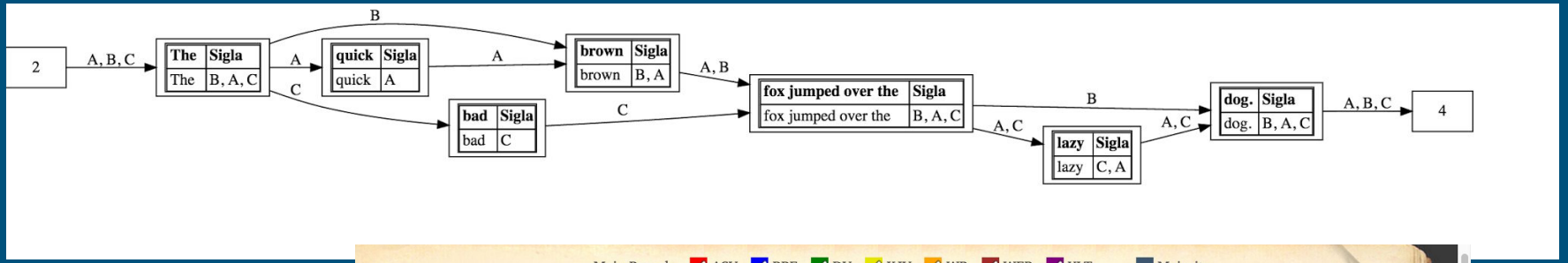
A	The	quick	brown fox jumps over the	-	dog.
B	The	-	brown fox jumps over the	lazy	dog.

A	B	C
The	The	The
quick	-	bad
brown	brown	-
fox	fox	fox
jumped	jumped	jumped
over	over	over
the	the	the
lazy	-	lazy
dog	dog	dog
.	.	.

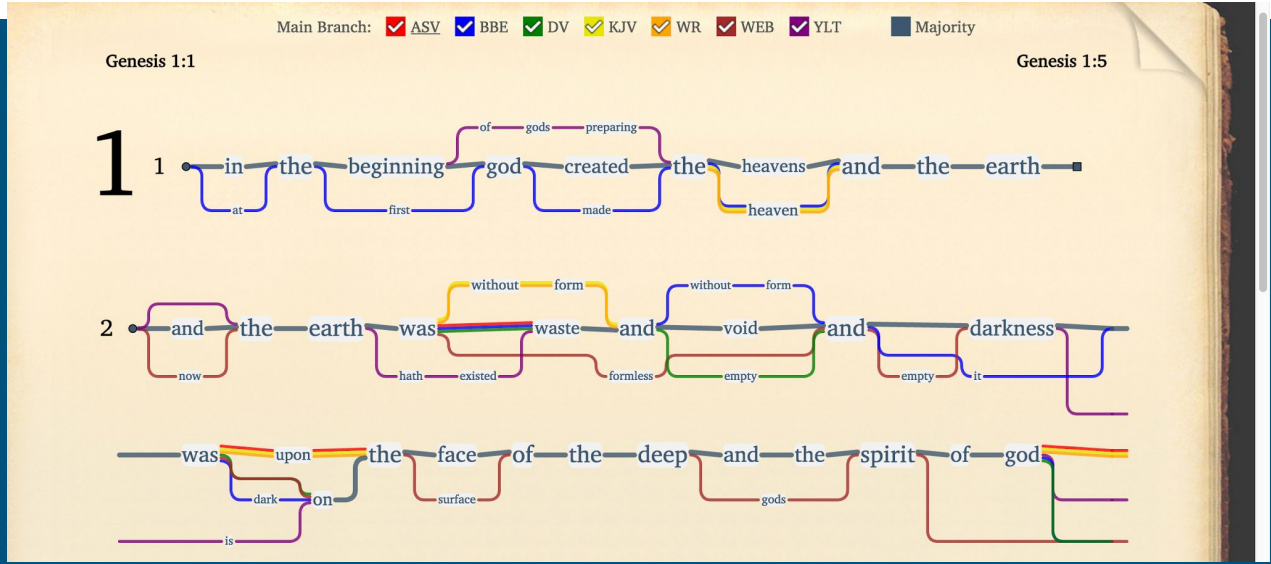
Alignment Table

W1	The black			cat
W2	The black	and	white	cat
W3	The black	and	green	cat
W4	The black		very special	cat
W5	The black	not	very special	cat

As a graph



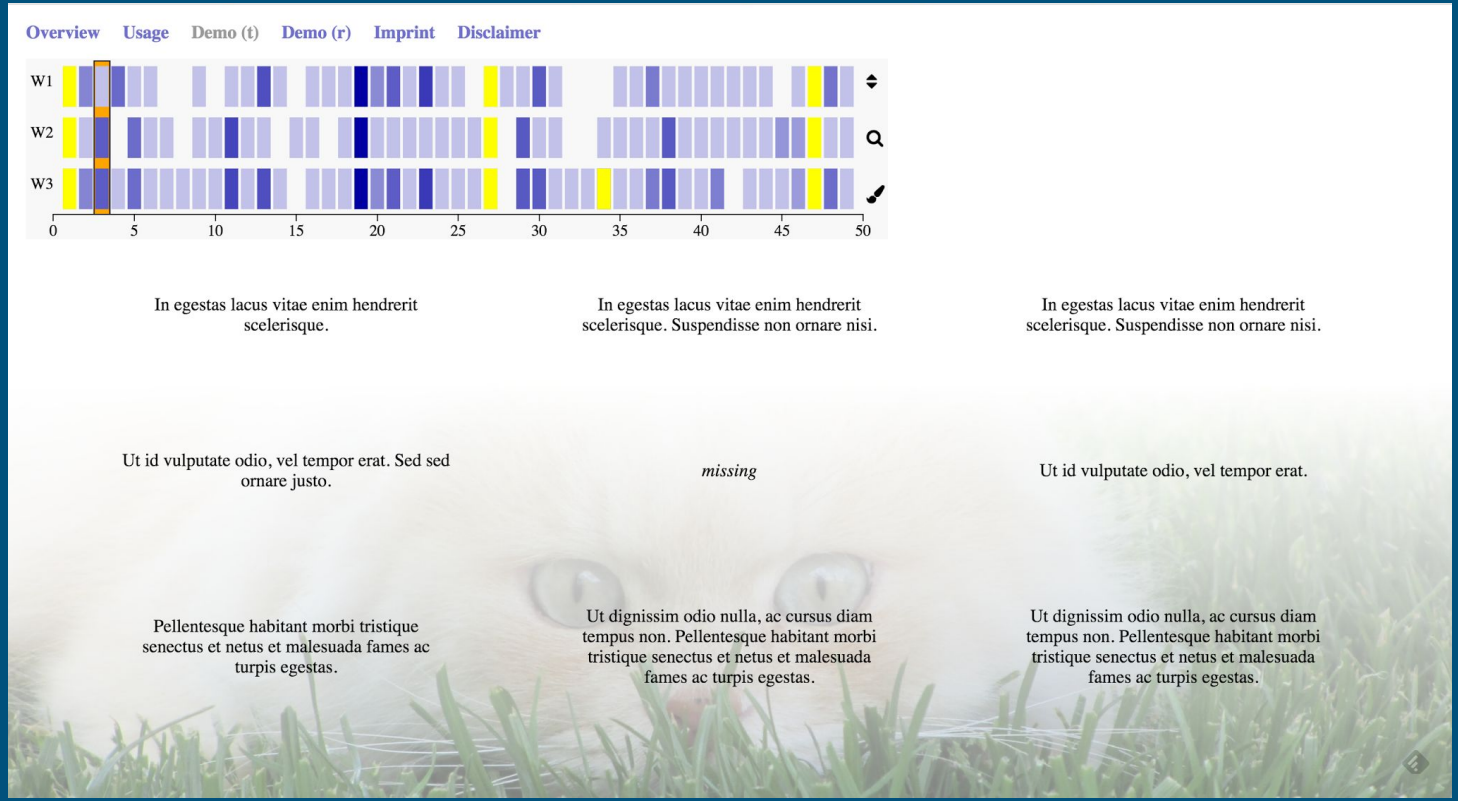
Alignment graph using CollateX (http://collatex.net/)



Alignment graph using TRAViz (http://www.traviz.vizcovery.org/)

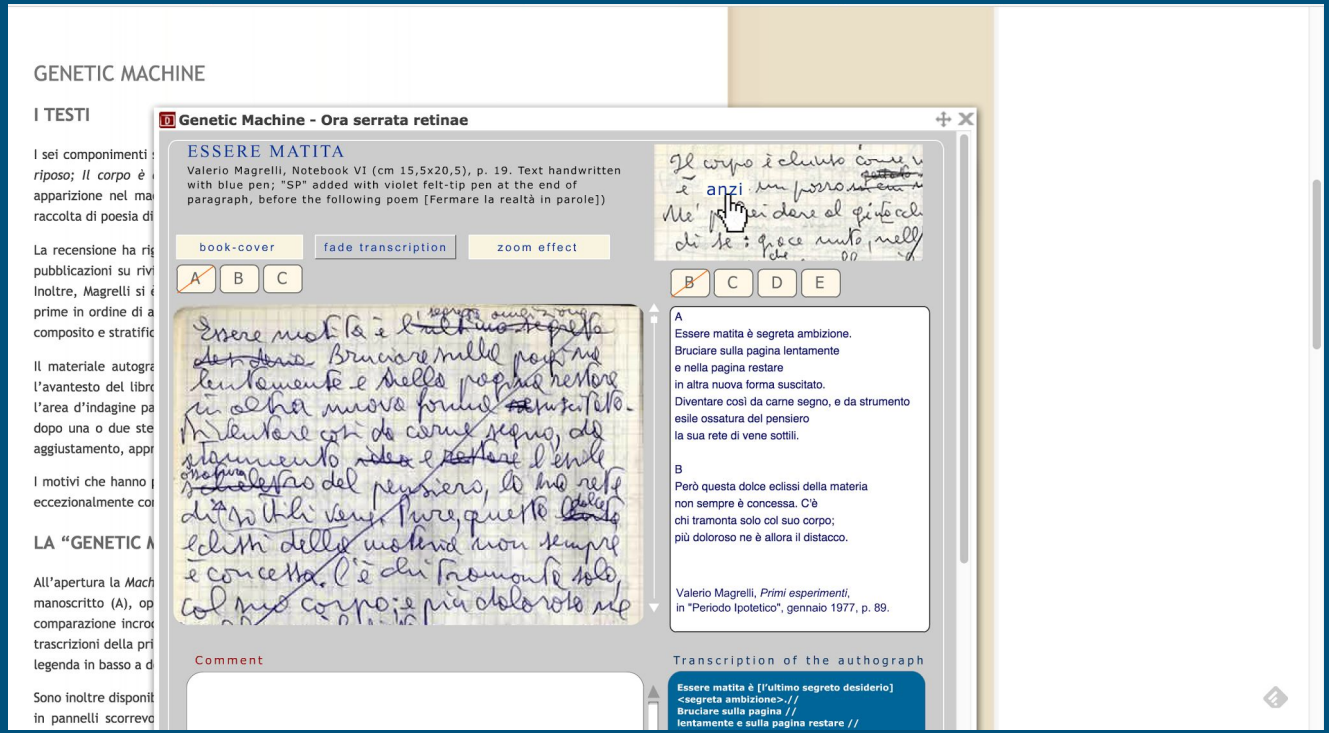
As matching segments in aligned sentences

Alignment of three sample texts on CATView (http://catview.uzi.uni-halle.de/overview.html)



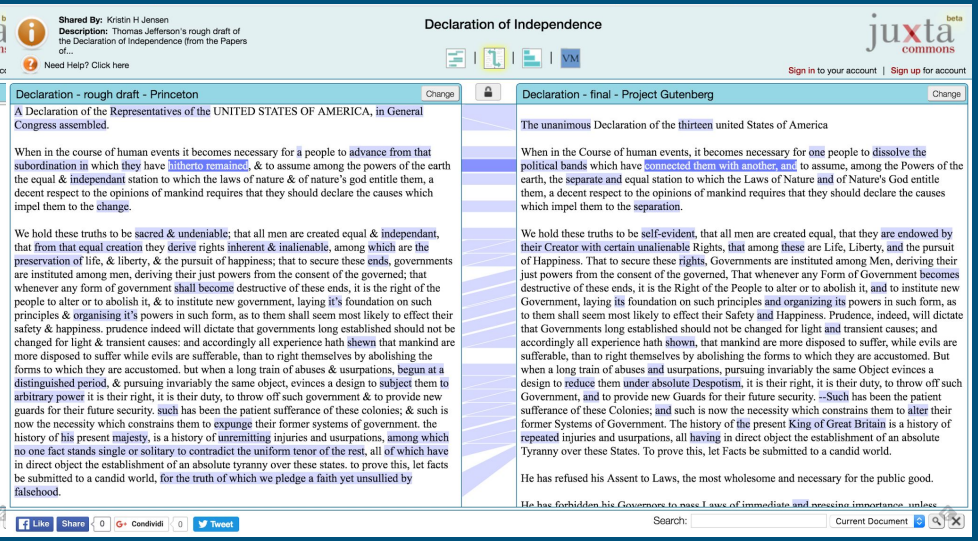
As a dynamic visualization

(http://www.digitalvariants.org/variants/valerio-magrelli)



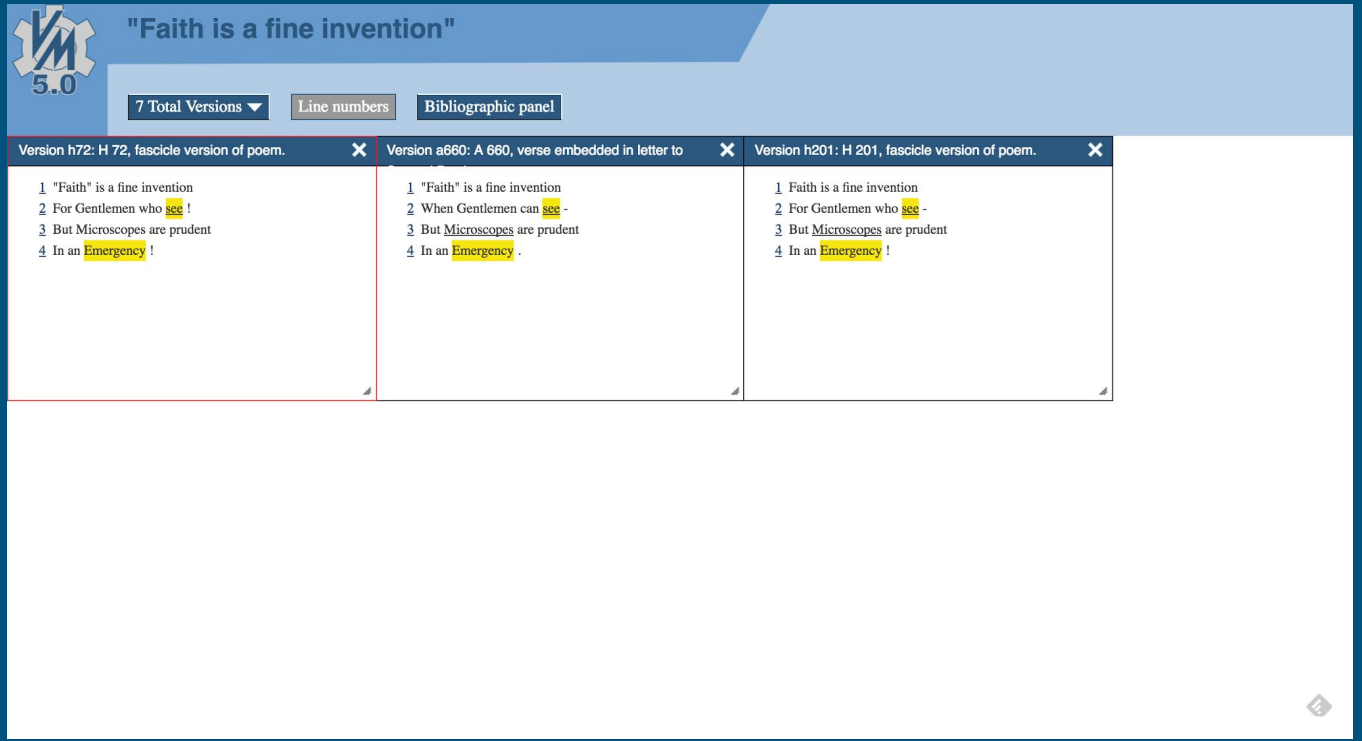
As overlapping variants (http://juxtacommons.org/)

As parallel texts with variants highlighted in the corresponding sections (http://juxtacommons.org/)



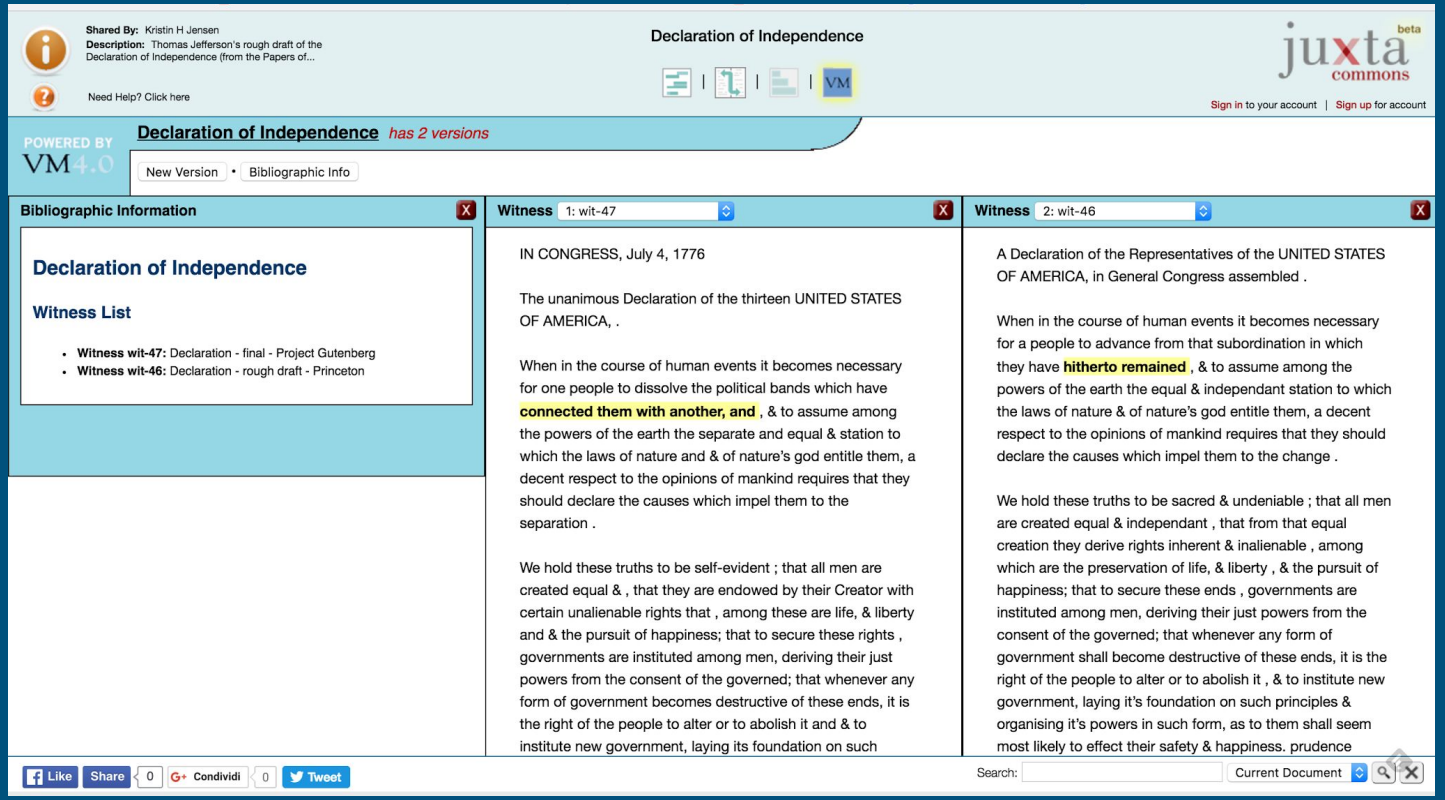
Why do we align texts?

To highlight correspondences in different versions of a text



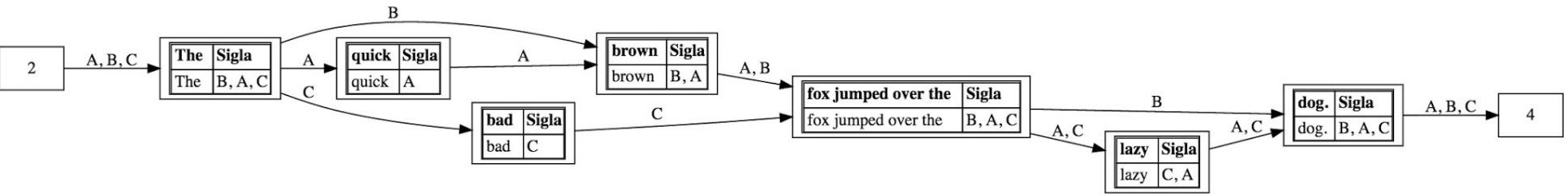
(http://v-machine.org/samples/faith.html)

To highlight divergences across various versions of the same text



(http://juxtacommons.org/)

```
+---+-----+-------+-------------------------+------+------+
| A | The | quick | brown fox jumps over the | -    | dog. |
| B | The | -     | brown fox jumps over the | lazy | dog. |
+---+-----+-------+-------------------------+------+------+
```



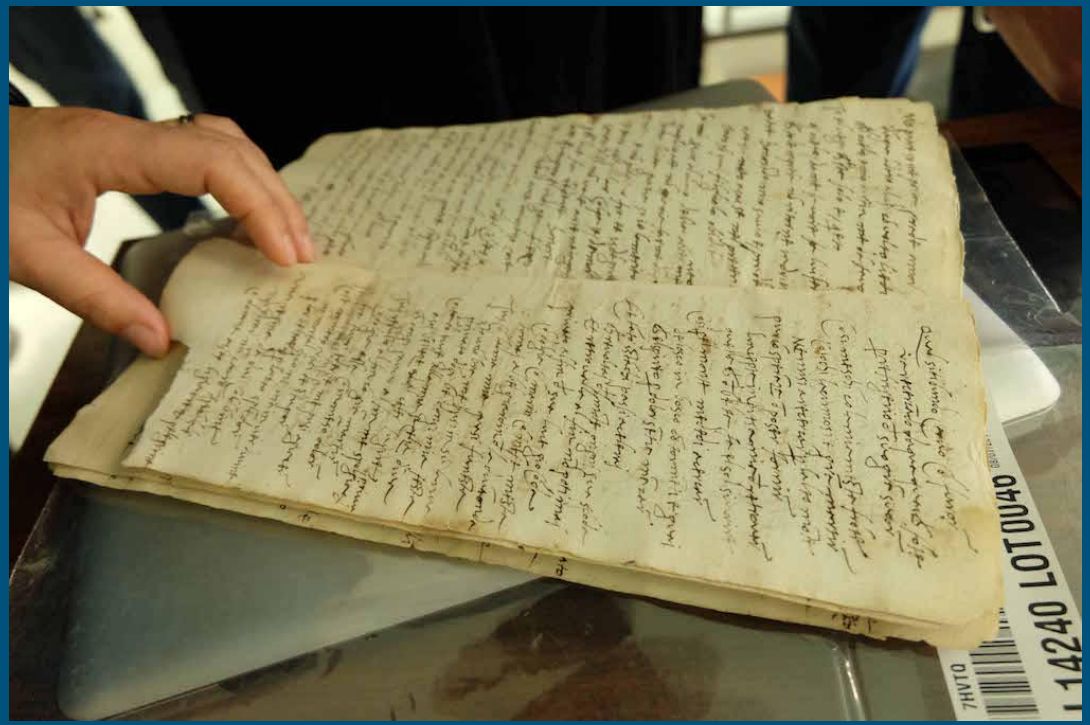
To establish relations between witnesses of a text and see where they overlap and diverge

Comparing texts as philological practice

Collatio

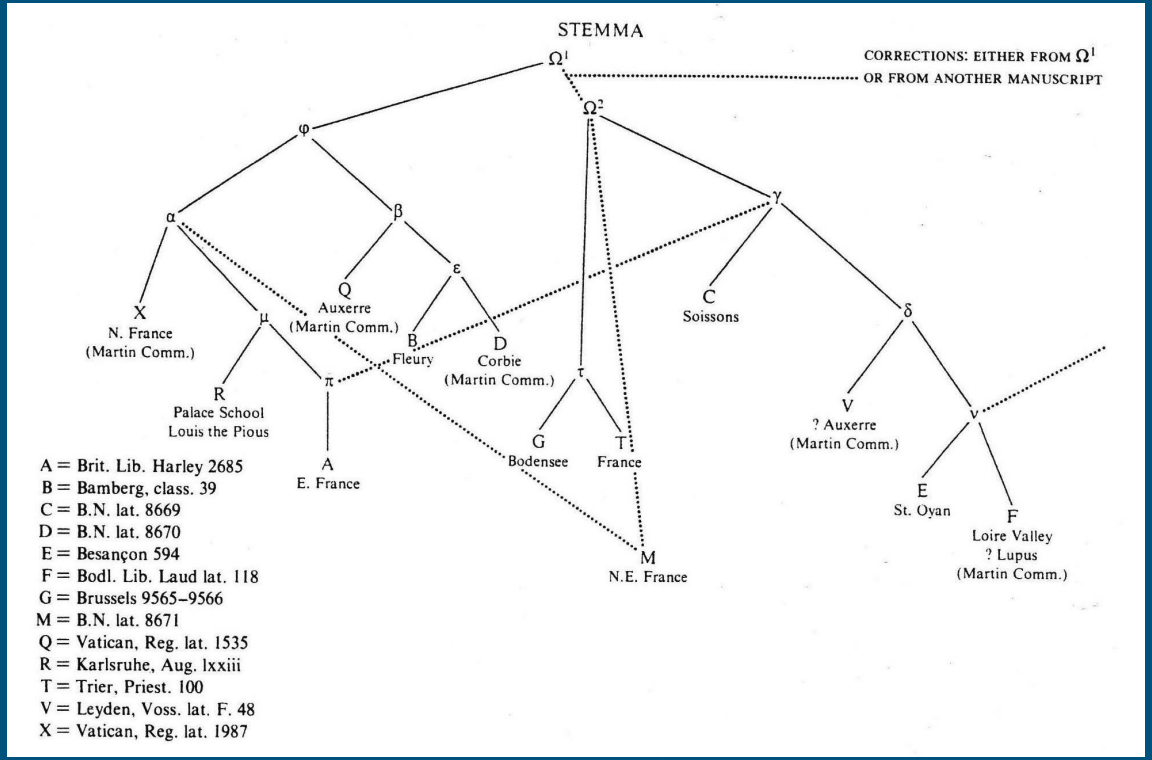
- Detection and transcription of variants in witnesses
- It is made by close reading each witness and comparing the texts with each other
- Evaluation of the variants and of the witnesses bearing them

....and yes, it is usually done manually.



Recensio

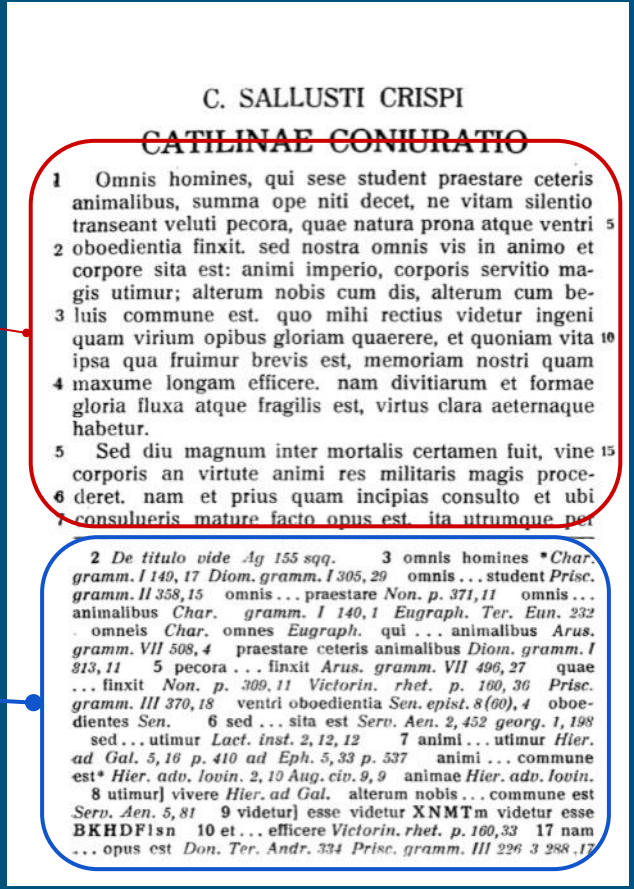
- To establish relationships between witnesses and which ones bear the “best text”
- To establish an organic scheme the transmission of a text, often represented as a genealogical tree of witnesses (*stemma*)



Example of a stemma. Stemma for *De nuptiis Philologiae et Mercurii* by Martianus Capella proposed by Danuta Shanzer (1986, p. 62-81).

Critical editions

- Usually display textual variants in the form of *apparatus criticus*
- The *apparatus* is a choice in itself: it does not collect all the variants found through collation, but only those that the editor had judged significant for the reconstruction of the text
- The *apparatus* can be very complex to understand in large textual traditions

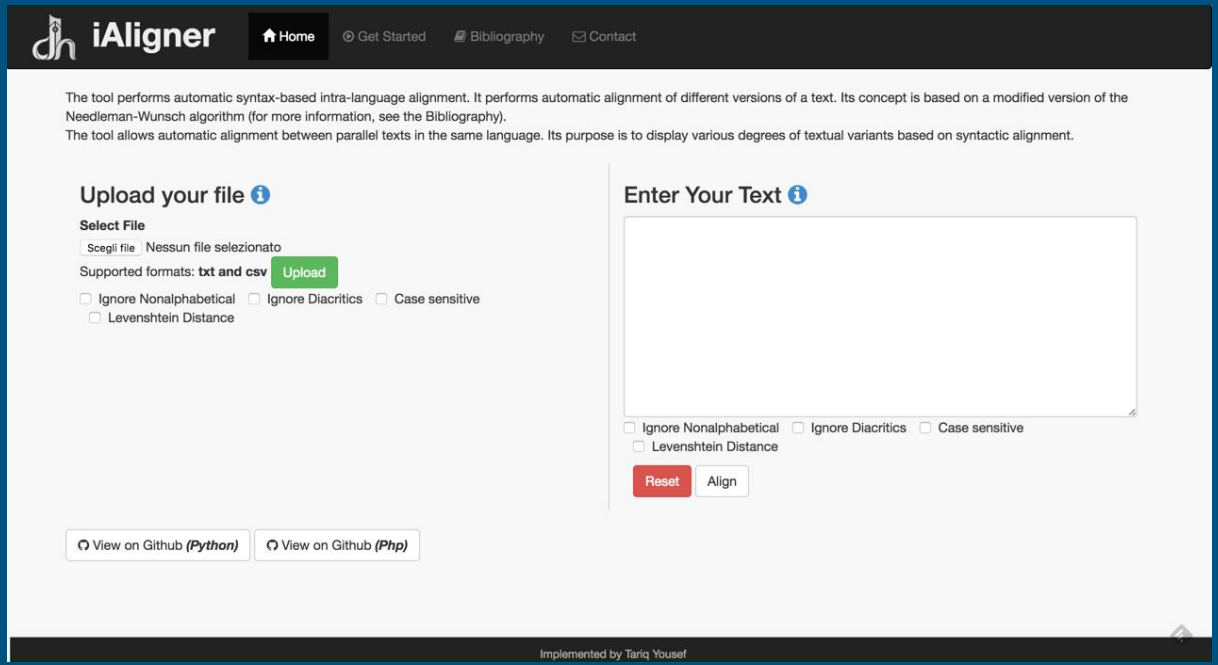


Sallust's *Catiline* in Axel Ahlberg's 1919 Editio Major.

Now we can do some of these things automatically

iAligner

http://i-alignment.com/
https://github.com/OpenGreekAndLatin/ILA_python



A tool for automatic syntax-based intra-language alignment

- **Automatic:** it is performed with algorithmic methods to reduce human intervention in the mechanical process of comparison.
- **Syntax-based:** in programming language, defines the order of the characters and the order of the words in a sentence.
- **Intra-language:** works with texts in the same language.
- **Pairwise or multiple:** works with two texts or with an unlimited number of multiple texts.

Algorithmic methods to produce alignment

The Needleman-Wunsch algorithm

- used in bioinformatics to align protein or nucleotide sequences.
- it uses Dynamic Programming to find the optimal alignment.
- divides a large problem into a series of smaller problems and uses the solutions to the smaller problems to reconstruct a solution to the larger problem.
- uses a score function and similarity matrix to represent all possible combinations of tokens and their resulting score.

The Needleman-Wunsch algorithm

- Aligning Bible Text John 1:1

NLT: In the beginning the Word already existed.

KJB: In the beginning was the Word

$$M[0][j] = j * indel \quad where \quad j \epsilon [0, n],$$
$$M[i][0] = i * indel \quad where \quad i \epsilon [0, m]$$

$$M[i][j] = max \begin{cases} M[i-1][j-1] + m \searrow & Matching \\ M[i-1][j-1] + mis \searrow & Mismatching \\ M[i][j-1] + indel \downarrow & Gap\ in\ S1 \\ M[i-1][j] + indel \rightarrow & Gap\ in\ S2 \end{cases}$$

$$Where \qquad 0 < i <= m \ , \ 0 < j <= n$$

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓								
the	-4↓								
beginning	-6↓								
was	-8↓								
the	-10↓								
Word	-12↓								
,	-14↓								

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓	**5** ↘	3→	1 →	-1 →	-3 →	-5 →	-7 →	-9 →
the	-4↓	0↓							
beginning	-6↓	-2↓							
was	-8↓	-4↓							
the	-10↓	-8↓							
Word	-12↓	-10↓							
,	-14↓	-12↓							

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓	5 ↘	3→	1 →	-1 →	-3 →	-5 →	-7 →	-9 →
the	-4↓	0↓	10 ↘	8 →	13 →	11 →	9 →	7 →	-5 →
beginning	-6↓	-2↓	8 ↓						
was	-8↓	-4↓	6 ↓						
the	-10↓	-8↓	4 ↓						
Word	-12↓	-10↓	2 ↓						
,	-14↓	-12↓	-0 ↓						

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓	5 ↘	3→	1 →	-1 →	-3 →	-5 →	-7 →	-9 →
the	-4↓	0↓	10 ↘	8 →	13 →	11 →	9 →	7 →	-5 →
beginning	-6↓	-2↓	8 ↓	15 ↘	13 →	11 →	9 →	7 →	-5 →
was	-8↓	-4↓	6 ↓	8 ↓					
the	-10↓	-8↓	4 ↓	5 ↓					
Word	-12↓	-10↓	2 ↓	0 ↓					
,	-14↓	-12↓	-0 ↓	-5 ↓					

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓	5 ↘	3→	1 →	-1 →	-3 →	-5 →	-7 →	-9 →
the	-4↓	0↓	10 ↘	8 →	13 →	11 →	9 →	7 →	-5 →
beginning	-6↓	-2↓	8 ↓	15 ↘	13 →	11 →	9 →	7 →	-5 →
was	-8↓	-4↓	6 ↓	8 ↓	11↓	9→	7→	5→	3→
the	-10↓	-8↓	4 ↓	5 ↓	13↘	11→	9→	7→	5→
Word	-12↓	-10↓	2 ↓	0 ↓	11↓	18↘	16→	14→	12→
,	-14↓	-12↓	-0 ↓	-5 ↓	9↓	16↓	14→	12→	10→

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

		In	the	beginning	the	Word	already	existed	.
	0	-2→	-4→	-6→	-8→	-10→	-12→	-14→	-16→
In	-2↓	5 ↘	3→	1 →	-1 →	-3 →	-5 →	-7 →	-9 →
the	-4↓	0↓	10 ↘	8 →	13 →	11 →	9 →	7 →	-5 →
beginning	-6↓	-2↓	8 ↓	15 ↘	13 →	11 →	9 →	7 →	-5 →
was	-8↓	-4↓	6 ↓	8 ↓	11↓	9→	7→	5→	3→
the	-10↓	-8↓	4 ↓	5 ↓	13↘	11→	9→	7→	5→
Word	-12↓	-10↓	2 ↓	0 ↓	11↓	18↘	16→	14→	12→
,	-14↓	-12↓	-0 ↓	-5 ↓	9↓	16↓	14→	12→	10→

The used score function (Matching = 5, Mismatching = -5, In/Del = -2)

The Needleman-Wunsch algorithm

John 1:1

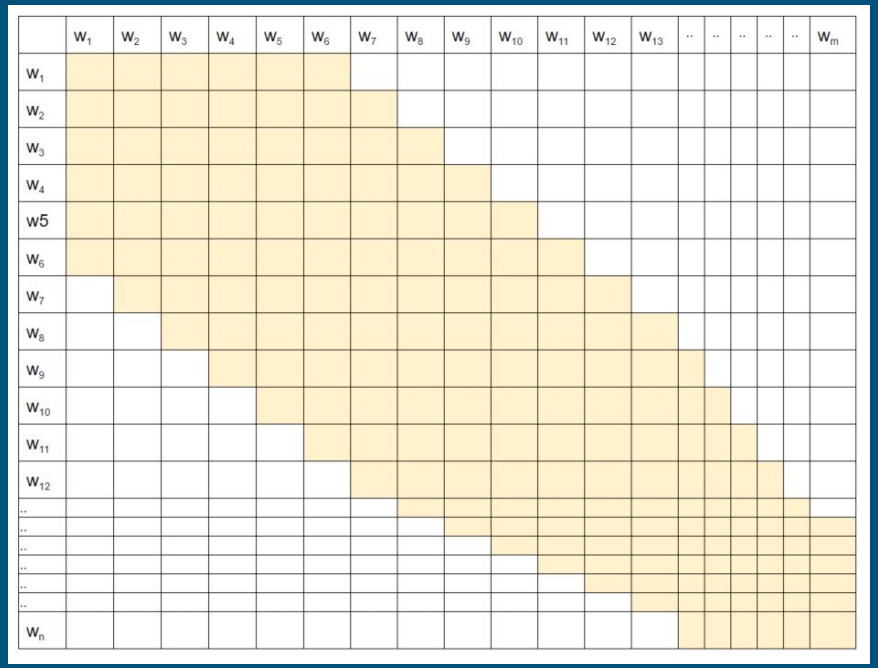
New Living Translation	In	the	beginning		the	Word	already	existed	.
King James Bible	In	the	beginning	was	the	Word			,

The modification to the algorithm

The goal is to optimize the algorithm by reducing the search space

compares a token `W` at the position i in `S1` with a range of tokens $[i-k, i+k]$ in `S2` with length of $2k+1$.

The resulting search space is reduced from $(n * m)$ to $([2k+1] * m)$, where $k < n/2$



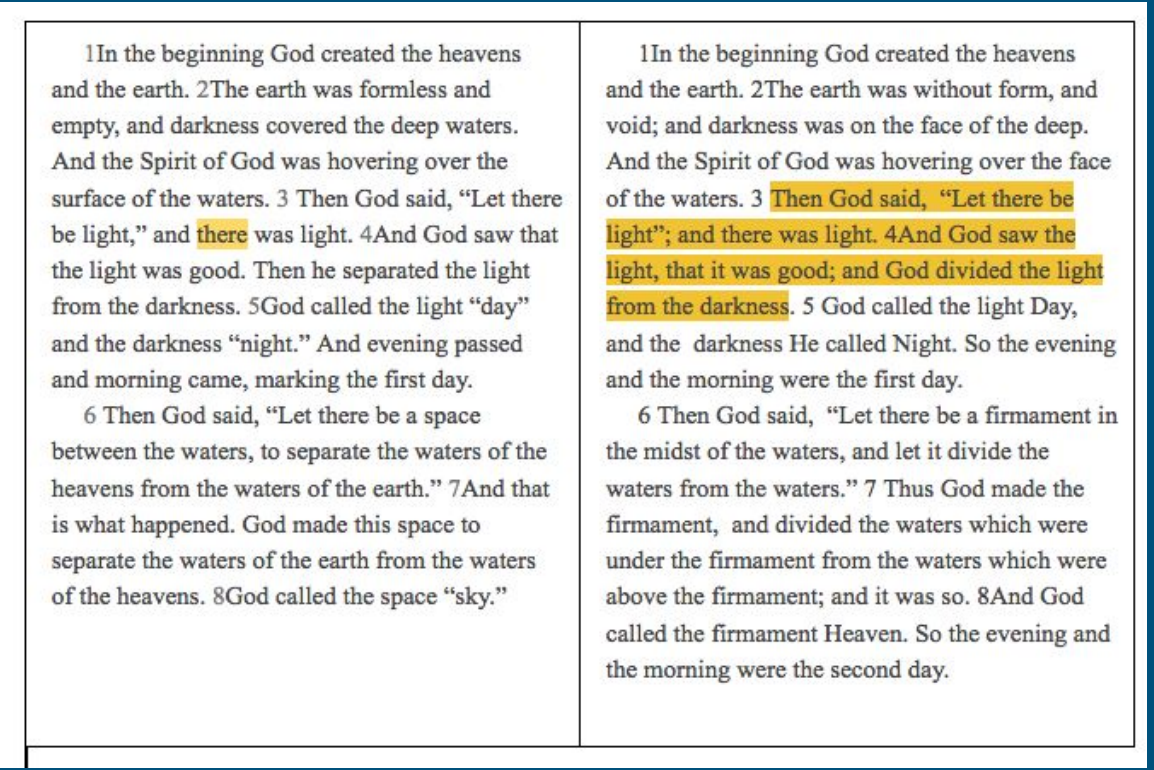
The modification to the algorithm

k = 14, n = 157, m = 134

*Search space = m*n = 21038*

after modification

*(2k+1)*m = 3886*



1In the beginning God created the heavens and the earth. 2The earth was formless and empty, and darkness covered the deep waters. And the Spirit of God was hovering over the surface of the waters. 3 Then God said, "Let there be light," and there was light. 4And God saw that the light was good. Then he separated the light from the darkness. 5God called the light "day" and the darkness "night." And evening passed and morning came, marking the first day.

6 Then God said, "Let there be a space between the waters, to separate the waters of the heavens from the waters of the earth." 7And that is what happened. God made this space to separate the waters of the earth from the waters of the heavens. 8God called the space "sky."

1In the beginning God created the heavens and the earth. 2The earth was without form, and void; and darkness was on the face of the deep. And the Spirit of God was hovering over the face of the waters. 3 Then God said, "Let there be light"; and there was light. 4And God saw the light, that it was good; and God divided the light from the darkness. 5 God called the light Day, and the darkness He called Night. So the evening and the morning were the first day.

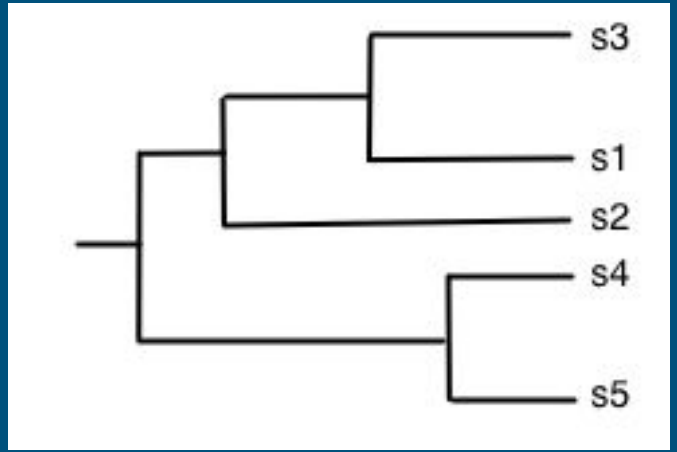
6 Then God said, "Let there be a firmament in the midst of the waters, and let it divide the waters from the waters." 7 Thus God made the firmament, and divided the waters which were under the firmament from the waters which were above the firmament; and it was so. 8And God called the firmament Heaven. So the evening and the morning were the second day.

Multiple Sequence Alignment (In progress)

- Progressive alignment

builds up a final MSA by combining pairwise alignments beginning with the most similar pair and progressing to the most distantly related, it requires two stages:

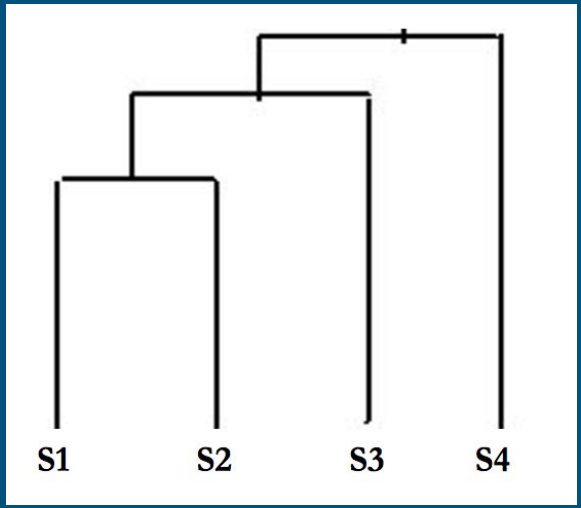
- creating the guide tree (clustering)
- adding the sequences sequentially to the growing MSA according to the guide tree



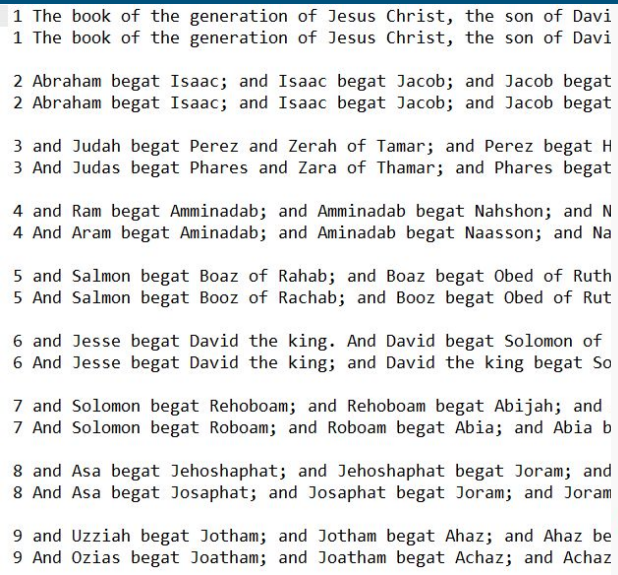
Multiple Sequence Alignment (In progress)

- Iterative alignment

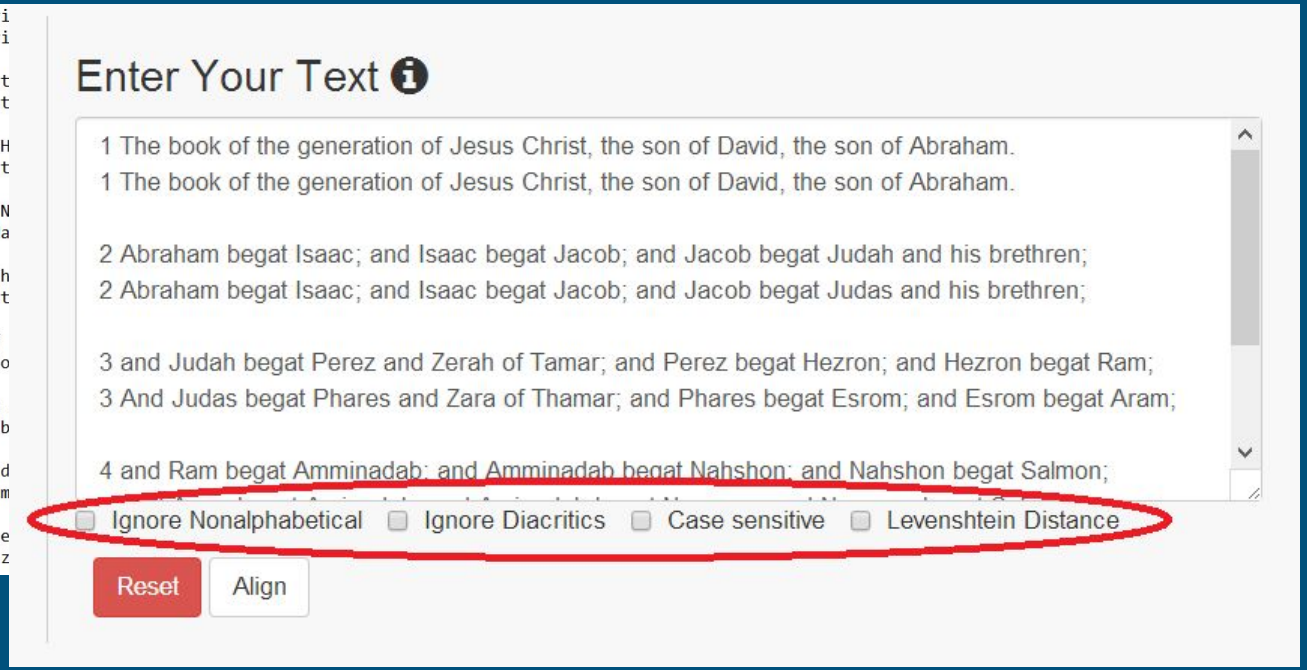
The aim is to reduce the problem of a multiple alignment to an iteration of pairwise alignments.



How to align your texts with iAligner: copy your text on the editor

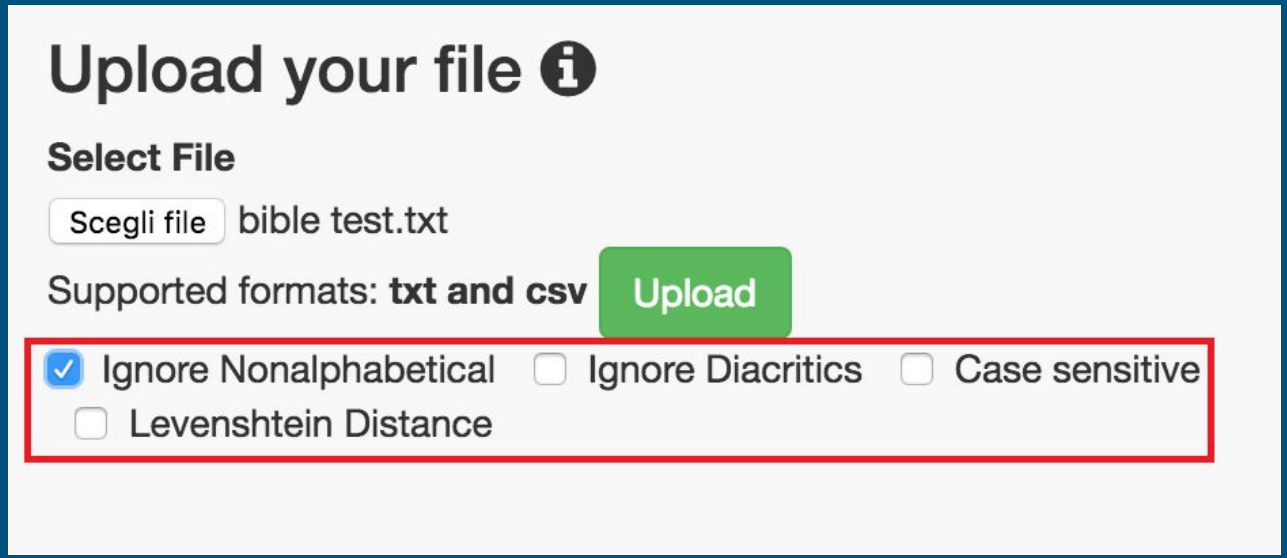


The text has to be parsed in sentences first

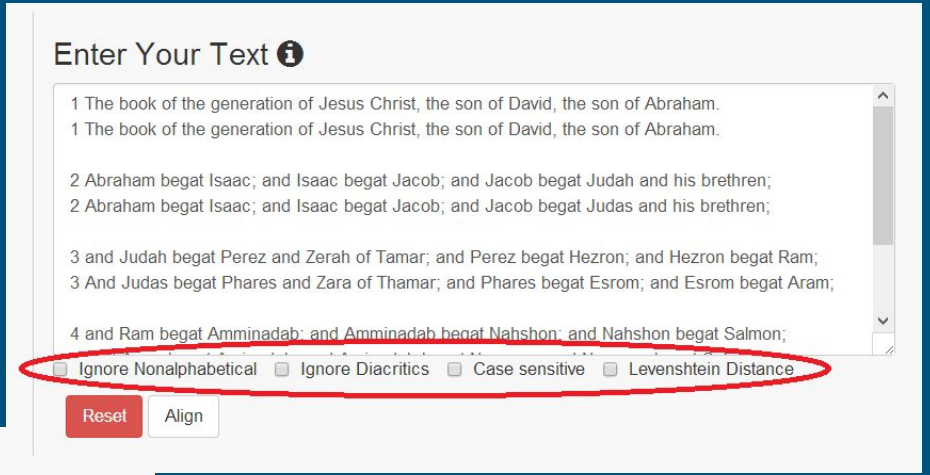


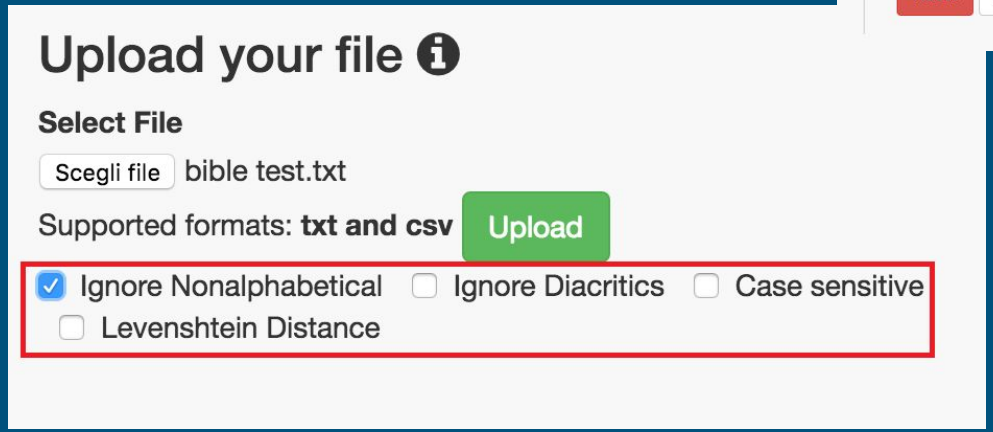
...Or upload it

Currently supports .txt and .csv files



Refinement criteria





- **Ignore non-alphabetical**: ignores symbols, such as punctuation and numbers, anything that is not an alphabetical character
- **Case sensitive**: if activated, detects variation across words according to the case
- **Ignore diacritics**: ignores any type of diacritical character (including punctuation marks)
- **Levenshtein distance**: applies a revised version of the Levenshtein algorithm and increases the tolerance threshold on the alignment of similar words.

The Levenshtein distance

The Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. e.g

```
lev(Hellanikos, Hellanicus) = 2
```

Mathematically, the Levenshtein distance between two strings { a,b} (of length |a| and |b| respectively) is given by $\mathrm{lev}_{a,b}(|a|, |b|)$

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Modified Levenshtein Distance

Levenshtein distance is not very helpful in our case, because it is binary and there is no tolerance with errors produced by OCR or Transcription.

the distance between letters is not binary, but it is on scale. The cost of insertion or deletion depends on:

- Letter position
- Letter type (vowel or consonant)

```
lev(Hellanikos, Hellanicus) = 0.3
```

Group elements	Distance within group elements
E, e Ē, ē I, i Y, y	0.15
Ō, ō O, o U,u Y, y	0.15
C, c K, k Q, q	0.15
S, s C, c	0.25
M, m N, n	0.35

A Greek text with no refinement criteria

The same text with additional refinement criteria applied

Alignment output: a table-graph

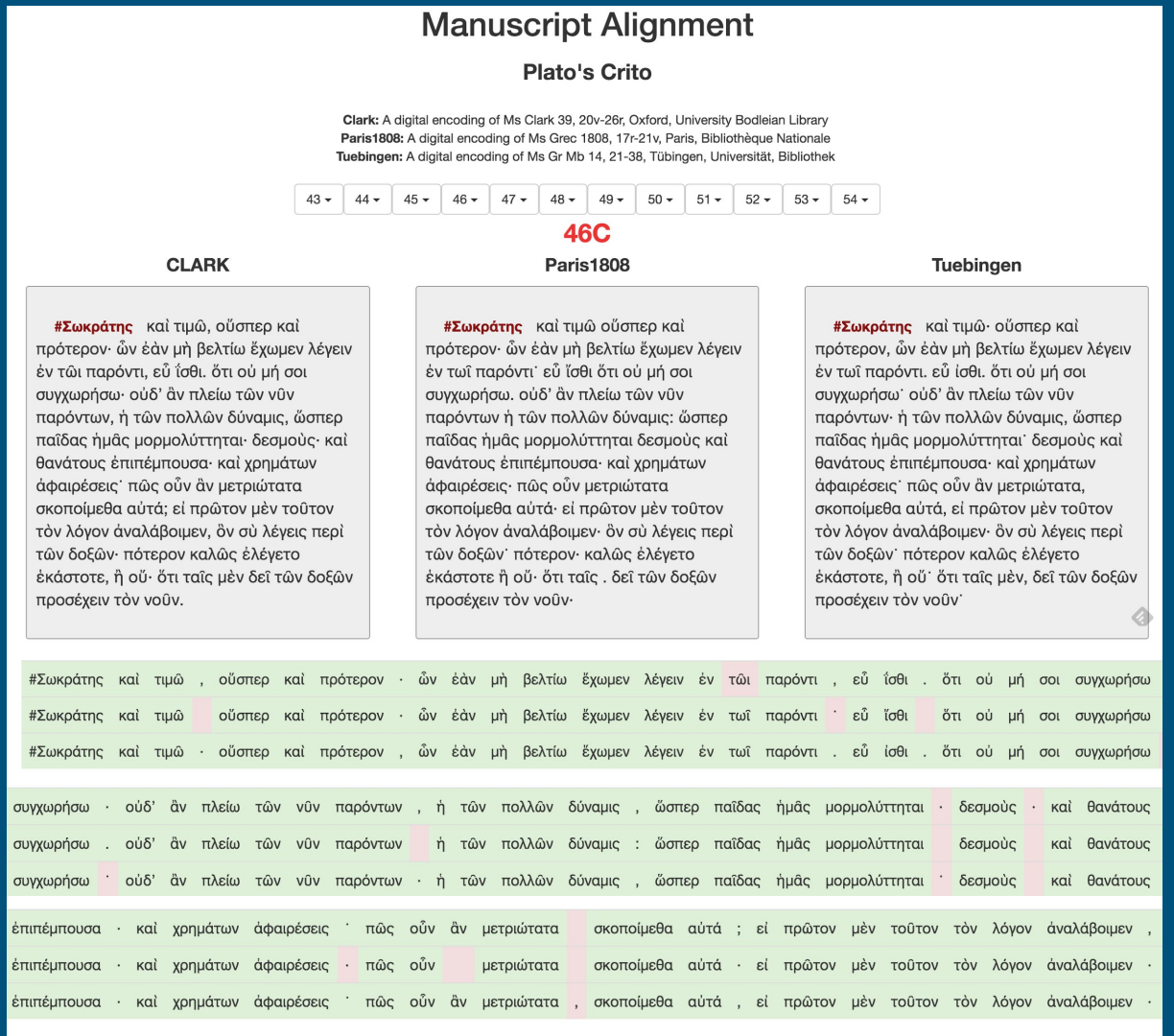
iAligner displays all the nuances of variants according to a color-key:

- Completely aligned tokens (deep green)
- Tokens aligned by excluding case sensitivity or punctuation detection (light green)
- Gaps (yellow)
- Divergences (red)
- Tokens aligned by applying Levenshtein distance (blue-green)

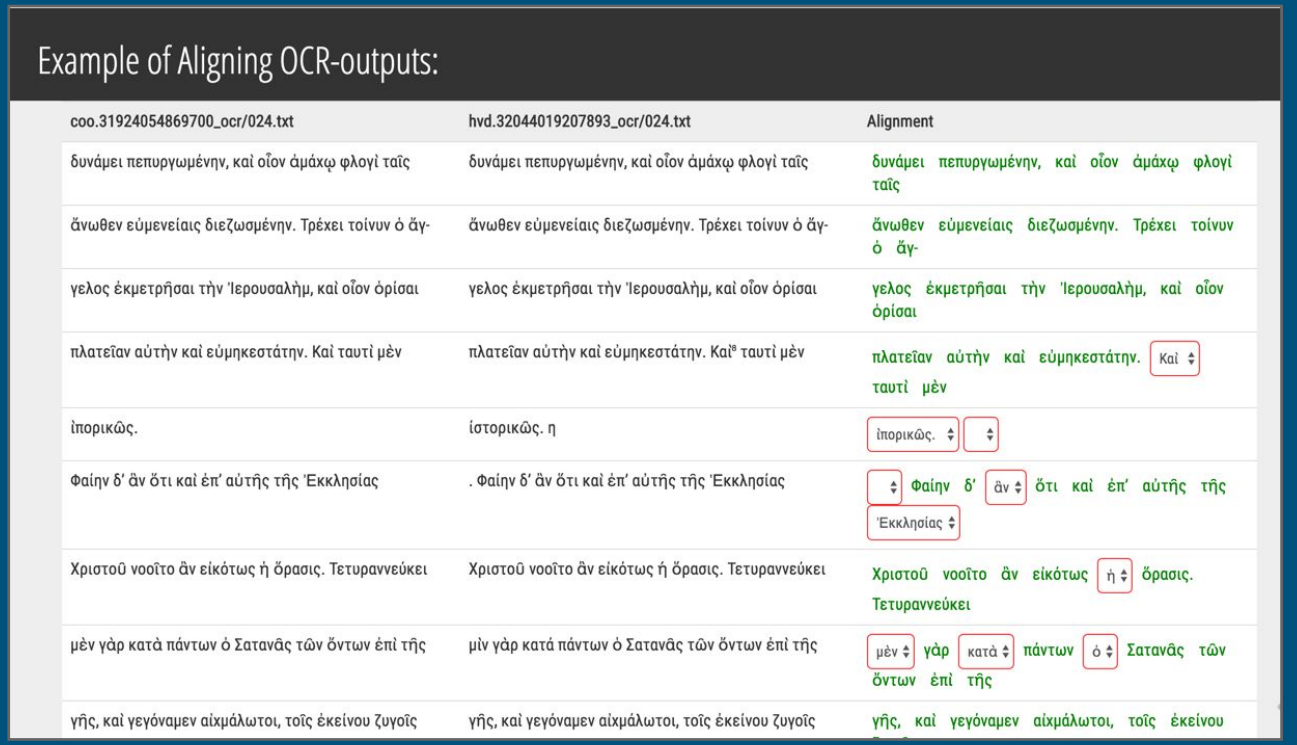
What can you do with iAligner? Some case studies

Manuscript alignment

Three manuscripts of Plato's *Crito aligned* (http://i-alignment.com/crito/)

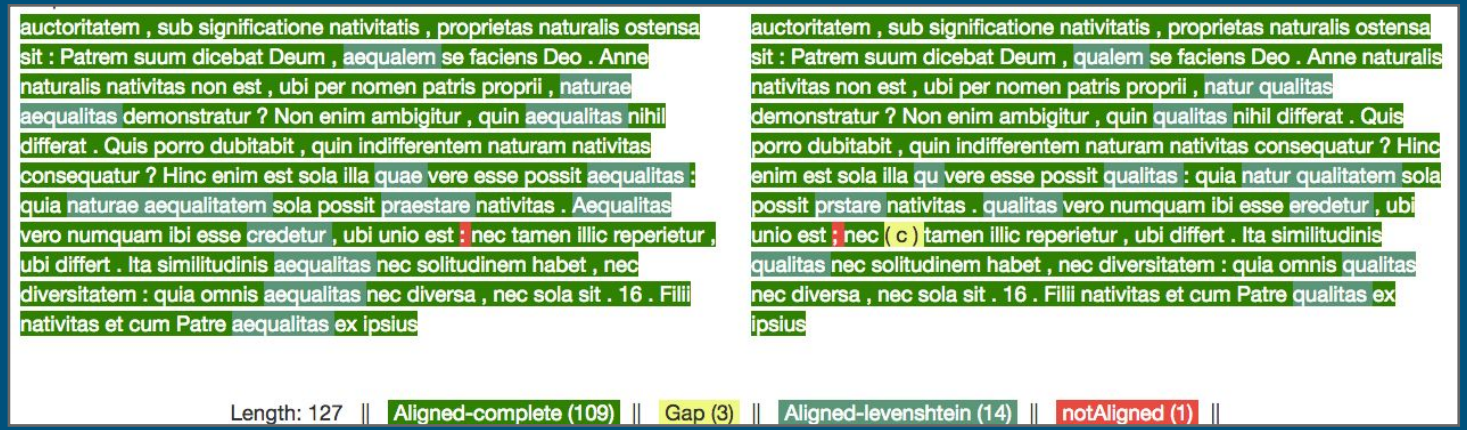


OCR output alignment



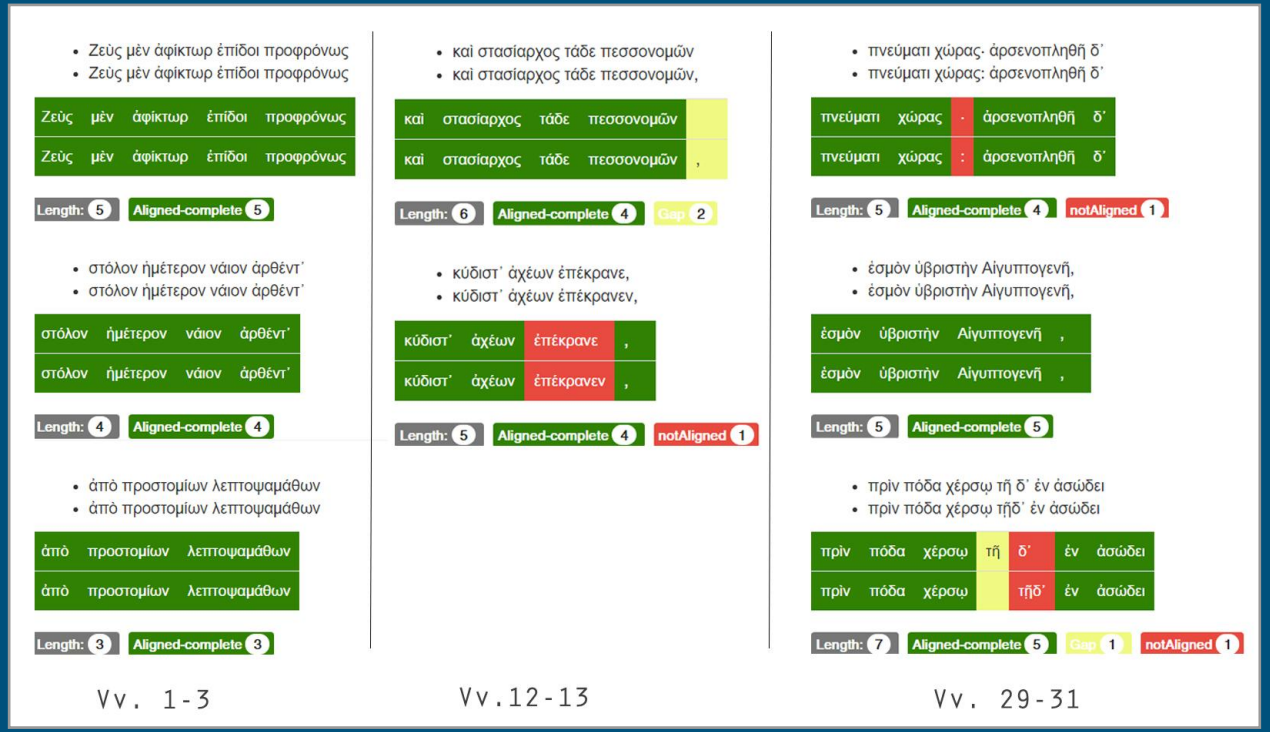
Alignment of two OCR outputs from the *Patrologia Graeca*. The third column shows the overlapping sections and offers the user the choice between two variants where the two texts diverge.

OCR output alignment



Patrologia Latina: OCR output vs. correct version: www.i-alignment.com/pl/

Alignment of editions



Three excerpted editions of Aeschylus' *Supplices* aligned. www.i-alignment.com/Aeschylus

Future work

Import and export options

Language dependent options for Latin, Greek, Arabic

Handling crossings and transpositions

Thanks for the attention!