Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

An Animated Demo of the Wagner-Fischer Algorithm for Sequence Alignment

Johann-Mattis List
Department of Linguistic and Cultural Evolution
Max Planck Institute for Evolutionary Anthropology

A long time ago I prepared an animated demo of the Wagner-Fischer algorithm for pairwise sequence alignment. Having used the demo to teach phonetic alignment in class, I thought it might be useful to share it officially, as it may also be interesting for colleagues who teach phonetic alignment or rudimentary JavaScript programming.

The Wagner-Fischer algorithm by Wagner and Fischer (1974) is less known as an alignment algorithm but more as the first algorithm that computes the Levenshtein distance (Levenshtein 1965). Conceptually, it is not much different from the Needleman-Wunsch algorithm by Needleman and Wunsch (1970). While the latter works with matching points between segment pairs, allowing for negative matches when segments are very different, the former works with penalties for divergent segments, setting segment similarity values as a rule to 0.

In my opinion, the Wagner-Fischer algorithm is much easier to understand, and it took me some time to appreciate the advantages of the Needleman-Wunsch algorithm, which lie mainly in its extensibility. Only the use of negative scores allows us to identify local maxima in the alignment of two sequences (Kondrak 2002). As a result, the Needleman-Wunsch algorithm can be easily extended to identify local alignments between two sequences (Smith and Waterman 1981), while this is not possible with the Wagner-Fischer algorithm.

The major idea of the Wagner-Fischer algorithm is to represent the comparison of two sequences by a matrix in which the first sequence is represented by the rows and the second sequence is represented by the columns. An alignment is a path through the matrix from the first top-left cell to the last bottom-right cell. To infer this path, the algorithm cumulatively computes all possible alignments between two sequences and then traces the path through the matrix back by choosing the path that minimizes the

Computer-Assisted Language Comparison in Practice
Volume 5, Number 3
https://calc.hypotheses.org/3265/

Published on 03/03/2022

penalties induced by the insertion of gaps and by mismatching segments (see List 2014 and Kondrak 2002 for detailed descriptions of the algorithm).

While it is quite easy and straightforward to understand how the algorithm works when seeing it in action, I have experienced myself how difficult it is to understand the algorithm when only reading the description. In fact, I would have never been able to understand the algorithm from the description I have given above. I managed to understand it, however, when I started to program it myself, introducing debug statements that would show me the resulting matrix during each step of comparison.

When I wanted to teach my students how sequences are aligned algorithmically (List 2015), my own difficulties in understanding the algorithm from text descriptions and sparse illustrations inspired me to make a life demonstration in which all major aspects of the algorithm are shown during each time step, so that one may even understand the algorithm from watching the demo alone. Thanks to JavaScript, creating this demo did not even take too much time.

The result is shown below. To get started, you need to insert two sequences in the two input fields and then click on the GO button. You can also stop the iteration or change its tempo.

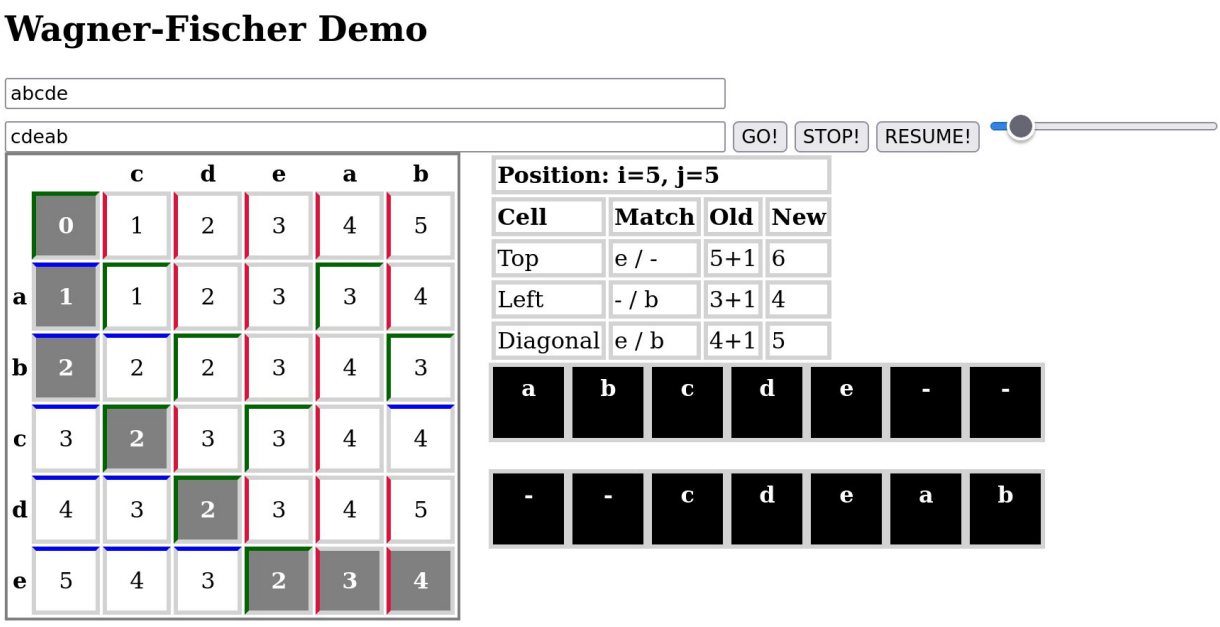

Figure 1: Wagner-Fischer Demo (see http://jsfiddle.net/LinguList/o8nbL07s/)

In the demo itself, three different aspects of the algorithm are shown: the alignment matrix, which is filled, step by step, the position of the algorithm in the matrix along with information on the match and penalty status, with the three choices, of which the one with the minimal penalty has to be chosen, and the resulting alignment of both sequences at this time step. When you hover over a cell highlighted in gray in the matrix (which marks it as part of the traceback, the path that indicates the alignment of both sequences),

you will see the current state of the alignment, in that the cell tells you which segments of both sequences it matches (either it matches both segments, or it introduces a gap in either of them).

I have shared the demo now in the form of a JSFiddle (http://jsfiddle.net/LinguList/o8nbL07s/), which shows both the original source code in JavaScript, the CSS (which is marginal), and the HTML file, and allows interested readers to experiment with the code and modify it.

References

Kondrak, Grzegorz (2002): Algorithms for language reconstruction . . University of Toronto: Toronto:.

Levenshtein, V. I. (1965): Dvoičnye kody s ispravleniem vypadenij, vstavok i zameščenij simvolov [Binary codes with correction of deletions, insertions and replacements]. Doklady Akademij Nauk SSSR 163.4. 845-848.

List, Johann-Mattis (2014): Sequence comparison in historical linguistics. Dusseldorf:Dusseldorf University Press.

List, Johann-Mattis (2015): Computergestutzter Sprachvergleich mit Python und Javascript [Computer-assisted language comparison with Python and Javascript]. Institut fur Sprache und Information: Heinrich Heine Universitat Dusseldorf.

Needleman, Saul B. and Wunsch, Christan D. (1970): A gene method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48. 443-453.

Smith, T. F. and Waterman, M. S. (1981): Identification of common molecular subsequences. Journal of Molecular Biology 1. 195-197.

Wagner, Robert A. and Fischer, Michael J. (1974): The string-to-string correction problem. Journal of the Association for Computing Machinery 21.1. 168-173.