Multiple sequence alignment in historical linguistics

A sound class based approach

Johann-Mattis List

In this paper, a new method for multiple sequence alignment in historical linguistics is presented. The algorithm is based on the traditional framework of progressive multiple sequence alignment (cf. Durbin et al. 2002:143-149) whose shortcomings are further enhanced by (1) a sound class representation of phonetic sequences (cf. Dolgopolsky 1986, Turchin et al. 2010) accompanied by specific scoring functions, (2) the modification of gap scores based on prosodic context, (3) a new method for the detection of swapped sites in already aligned sequences.

The algorithm is implemented as part of the LingPy library (http://lingulist.de/lingpy), a suite of open source Python modules for various tasks in quantitative historical linguistics. The method was tested on a benchmark dataset of 152 manually edited multiple alignments covering data for 192 Bulgarian dialects (Prokić et al. 2009). The results show that the new method yields alignments which differ only in 5 % of all sequences from the gold standard.

1. Sequences

Many structures we are dealing with – be it in daily life or in science – can be represented as *sequences*. The bird songs which awake us in the morning are sequences of sound waves, the movies we watch are sequences of pictures, and the meals we cook are created by a sequence of instructions received from a recipe book.

What recipes, movies, and music have in common, or – to put it in other terms – what allows us to view them as sequences, is that they all can be seen as ordered chains of objects whose identity is a product of both their *order* and their *content*.

Definition 1 Given an *alphabet* (a non-empty finite set, whose elements are called *characters*), a *sequence* is an ordered list of characters drawn from the alphabet. The elements of sequences are called *segments* (cf. Böckenbauer & Bongartz 2003:30f).

Proceedings of ConSOLE XIX, 2012, 241-260
http://www.sole.leidenuniv.nl

According to Definition 1, the constitutive elements of sequences are the *alphabet*, the *characters*, and the *segments*. One can imagine a sequence as a string of colored beads. If we take the beads separately from the string, it is impossible to distinguish those beads which have the same color from each other, as it is impossible to distinguish identical *characters* drawn from the same *alphabet*. Yet lining them up on a string, every bead becomes an individual *segment*, since it has a position different from all the other beads on the string (see Figure 1).

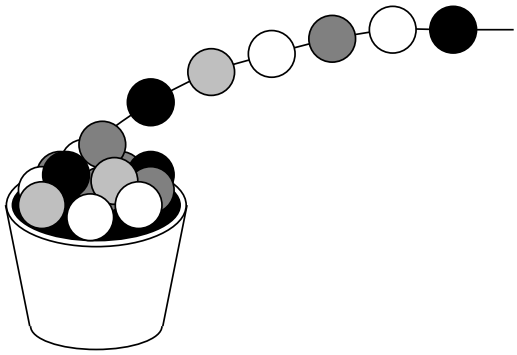

Figure 1: Sequences as strings of colored beads

2. Alignment analyses

Due to their complex character, the comparison of sequences has to be based on a comparison of both their segments and their structure. Comparing two sequences of which we assume that they are in a certain relationship, it is important to determine how (or if) the segments of the sequences correspond to each other. Ignoring complex matches (where one segment of one sequence corresponds to multiple segments in another sequence), we can distinguish three kinds of segment correspondence: *matches* (the segments are identical), *mismatches* (the segments are not identical), *empty matches* (one segment corresponds to a null-segment, see Figure 2).

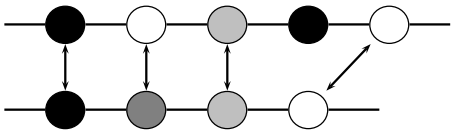

Figure 2: Sequence Comparison

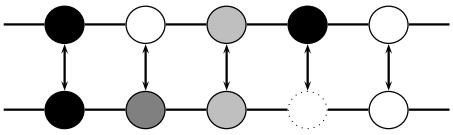

Figure 3: Alignment analyses

Alignment analyses are a specific way to model and to visualize differences in sequences. They reflect the three kinds of correspondences by replacing null-segments with gap characters. The result is a matrix in which all matching segments appear in the same column (see Figure 3).

Definition 2 An *alignment* of n sequences is an n-row matrix in which all sequences are arranged in such a way that all matching and mismatching segments occur in the same column, while empty cells, resulting from empty matches, are filled with gap symbols (cf. Gusfield 1997:216).

Alignment analyses are the most common way to model differences between sequences. With the help of alignment analyses all different kinds of sequences can be compared, regardless of where they occur or what the purpose of the comparison is. Thus, when trying to detect plagiarism in scientific work, texts can be interpreted as sequences and the words can be interpreted as their segments. In this way, an alignment analysis can shed light on the differences between the original text and the plagiary (see Example 1). In molecular biology, the alignment of protein and DNA sequences is a very common method and the basis of phylogenetic reconstruction (see Example 2), and in type setting programs and search engines, sequence alignments can be used to detect spelling errors (cf. Example 3).[1]

(1)

–	Ein	Motto	das	–	programmatisch	zu	verstehen	ist
und	dieses	Motto	–	ist	programmatisch	zu	verstehen	–

(2)

G	A	-	C	G	G	A	T	T	A	T	G
G	A	T	C	G	G	A	A	T	A	-	G

(3)

p	-	l	i	e	c	e	m	e	n
p	o	l	i	-	c	e	m	a	n

Since manually conducted alignments of sequences can be very time-consuming (especially in data-driven disciplines such as evolutionary biology) many different methods and algorithms for *automatic alignment analyses* have been proposed. When dealing with automatic alignment analyses, it is common to make a distinction between *pairwise sequence alignments* (PSA) and *multiple sequence alignments* (MSA). This is due to the fact that the computational solutions for pairwise alignment analyses which deal with the alignment of only two sequences differ greatly from multiple sequence analyses regarding their complexity. In the following, I shall roughly present the main ideas behind the most common algorithms for both pairwise and multiple sequence alignment.

2.1. Pairwise sequence alignment

The basic algorithm for computing an optimal alignment of two sequences was independently developed by different scholars from different scientific disciplines (cf., e.g., Wagner & Fischer 1974, Needleman & Wunsch 1970). An optimal global pairwise alignment of two sequencec is computed via a dynamic programming algorithm (DPA, cf. Gusfield 1997:217f). The basic idea of the algorithm is 'to build up an optimal alignment using

[1]Source text and plagiary in Example 1 are taken from Kommission "Selbstkontrolle in der Wissenschaft" der Universität Bayreuth 2011:Appendix 3, Example 2 is taken from Böckenbauer & Bongartz (2003:79).

previous solutions for optimal alignments of smaller subsequences' (Durbin et al. 2002:19). In order to do so, all segments of the sequences are confronted with each other and with gap characters in a matrix. The algorithm then recursively calculates the total scores for all subsequences by filling the matrix from top to bottom and from left to right. Once the score for one subsequence is known, the score for a larger subsequence can be calculated. In each recursion step, a specific *scoring function* is employed in order to evaluate whether the segments should be matched with themselves, or with one of the gap characters. Once the matrix is filled, the value in the last cell of the matrix will yield the best score. In order to obtain the alignment of the sequences, a *traceback function* has to be applied in order to find the 'path of choices […] which led to this final value' (Durbin et al. 2002:19).

	h –	e –	a –	r –	t –
– h	h – h	e – – h	a – – h	r – – h	t – – h
– e	h – – e	e – – e	a – – e	e – – e	t – – e
– r	h – – r	e – – r	a – – r	r – – r	t – – r
– z	h – – z	e – – z	a – – z	r – – z	t – – z

0	1	2	3	4	t –
1	0	1	2	3	t – – h
2	1	0	1	2	t – – e
3	2	1	1	r – – r	t – – r
4	3	2	2	r – – z	t – – z

0	1	2	3	4	5
1	h h	1	2	3	4
2	1	e e	a –	2	3
3	2	1	1	r r	2
4	3	2	2	2	t z

Figure 4: The dynamic programming algorithm

Which scoring function one uses for the computation depends on the purpose of the respective alignment analysis. A very simple scoring function, which penalizes mismatches and gaps with 1 and identity matches with 0 (the Levenshtein metric, cf. Levenshtein 1966) is employed in the illustration of the algorithm in Figure 4. Here, the two strings `heart` and `herz` (cf. English *heart* and German *Herz* 'heart') are aligned. The first matrix shows how all segments of both sequences are matched with each other and with gap characters. The second matrix shows how the cost for every subsequence is calculated by summing up all scores recursively by starting from the first row of the first column, moving from top to down and from left to right. In each step, the scoring function evaluates three possible alignment pairs for the values in each cell. The second column in the second row for example is filled with 0 (which is the lowest cost and therefore the best score), since, according to the Levensthein scoring function, matching the two segments as $\binom{h}{h}$ is better than matching one of the segments with a gap yielding $\binom{h}{-}$ or $\binom{-}{h}$. The third matrix shows, how the alignment is obtained by a traceback.

2.2. Multiple sequence alignment

While the dynamic programming algorithm is useful for finding an optimal solution for the problem of pairwise sequence alignments, it is not practical for the computation of multiple sequence alignments, since the computational effort will increase enormously with the number of sequences being analyzed (Gusfield 1997:345). It is therefore common to employ certain heuristics which can only guarantee to find a near-optimal solution for

multiple sequence alignments. Among the most popular algorithms applied in multiple sequence analyses are the so-called progressive techniques (Feng & Doolittle 1987; Higgins & Sharp 1988; Thompson et al. 1994). These approaches start by constructing a *guide tree* from the pairwise alignment scores of all sequence pairs using traditional cluster algorithms such as UPGMA (Sokal & Michener 1958) or Neighbor-joining (Saitou & Nei 1987). This guide tree is then used to align all sequences successively with each other, moving from its branches down to its root. Figure 5 gives an example for progressive alignment based on a guide tree for the four sequences Czech *jablko* [jabl̩ko], Bulgarian *jabălka* [jabəlka], Russian *jabloko* [jabləkə], and Polish *jabłko* [japko] 'apple'.

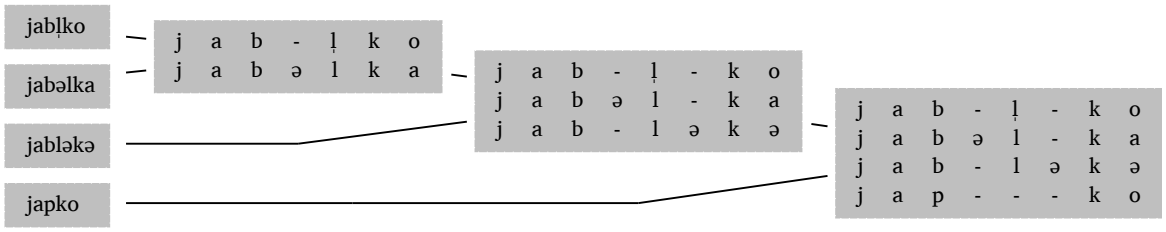

Figure 5: Progressive MSA

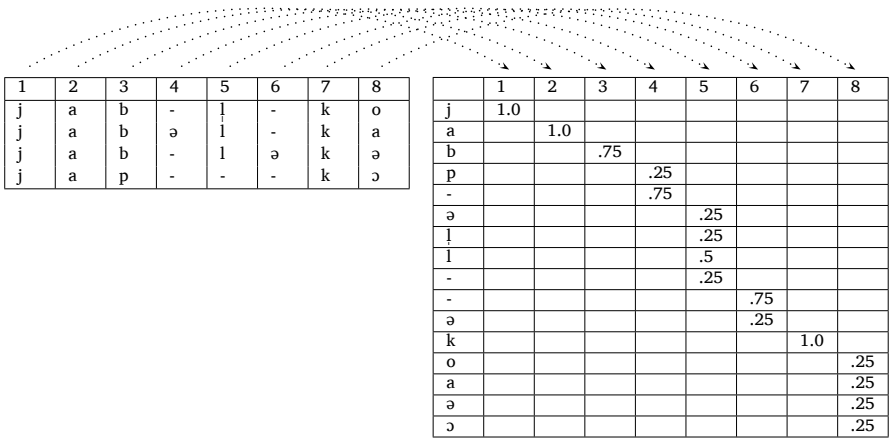

Figure 6: Profile representation of an MSA

The algorithm for progressive MSA can be further enhanced in different ways, the most important improvement being the application of *profiles*. A profile represents the relative frequency of all segments of an MSA in all its positions (Durbin et al. 2002:146f). A profile therefore can be seen as a sequence of vectors (see Figure 6). In profile-based approaches, once two sequences are aligned along the guide tree, they are further represented as profiles. When aligning already joined sequences, the traditional dynamic programming algorithm is used to align profiles with profiles, or profiles with sequences. The alignment score for the columns of two profiles is usually calculated as a *sum-of-pairs score* (Durbin et al. 2002:138f) which is the mean of the sum of the pairwise scores for all residue pairs in the two columns. The advantage of profile approaches over approaches which take one

sequence as representative for a whole multiple alignment is that more information can be taken into account during the alignment process.

2.3. The purpose of alignment analyses

When dealing with alignment analyses, it is of great importance to be aware of their potentials and limits. Alignments themselve do *not* draw an evolutionary scenario which explains how sequences evolved from common ancestor sequences into their current shape. Alignments merely tell us which segments of the aligned sequences share a common history, or – to state it in biological terms – which residues among a set of aligned sequences are *homologous* (Durbin et al. 2002:135). From the viewpoint of historical linguistics, alignment analyses – though seldom explicitly applied (with the only exceptions I am aware of being Anttila 1972:229-263) – are only the first stage of linguistic reconstruction.

3. Sequence comparison in historical linguistics

Sequence comparison in historical linguistics, i.e. the comparison of words and morphemes which have evolved from the same ancestor forms, is traditionally carried out manually. Yet the technique by which systematic correspondences between words from different languages are retrieved are not much different from the techniques which are applied in automatic alignment analyses in evolutionary biology. Figure 7 illustrates, how the systematic comparison of the words German *Zahn* [tsʰaːn], English *tooth* [tʊːθ], Italian *dente* [dɛnte] and French *dent* [dɑ̃] leads to a reconstruction of the proto stages of the languages.

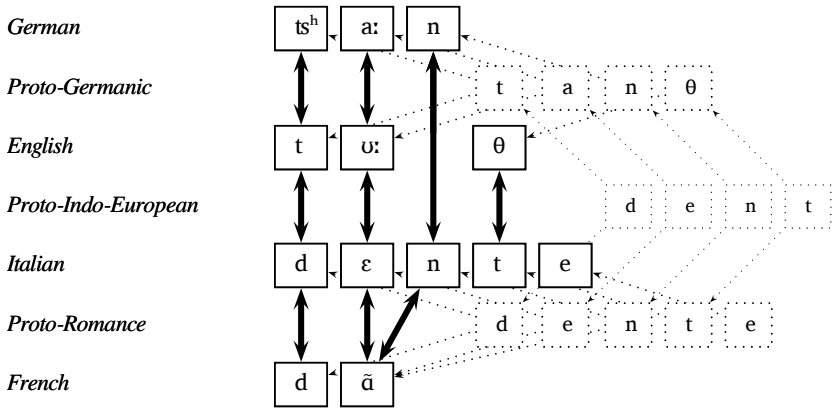

Figure 7: Structural comparison of multiple words

3.1. Sequence similarity

In linguistics one can distinguish two different perspectives regarding the similarity of phonetic sequences, which I will call the *synchronic* and the *diachronic* perspective. From a synchronic perspective sequences are judged to be similar, if they show resemblances regarding the way they are produced or perceived. From a diachronic perspective, however, sequences are judged to be similar, if they have a common origin. The words in Table 1 are similar from a synchronic viewpoint, since they consist of synchronically similar segments. The words in Table 2, however, are similar from a diachronic viewpoint, since they consist of segments which can be traced back to common ancestors.[2]

Greek	**mati**	'eye'	≈	Malay	**mata**	'eye'
Greek	**θɛɔs**	'god'	≈	Spanish	**diɔs**	'god'

Table 1: Synchronic similarity

German	**tsʰaːn**	'tooth'	≈	English	**tʊːθ**	'tooth'
Spanish	**etʃo**	'fact'	≈	French	**fɛ**	'fact'

Table 2: Diachronic similarity

The differences between the synchronic and the diachronic perspective lie in the way in which similarity is defined: Synchronic resemblance is determined by an evalution of phonetic or acoustic features. If the features of the segments are close, the segments are judged to be similar, and likewise the sequences. Diachronic resemblance between two sequences is determined by comparing the distinctive function which their segments fulfil in both language systems: If a comparison of the languages shows that 'whenever word W1 in language L1 contains sound S1 in a certain position, then a word W2 of the same or similar meaning in language L2 contains sound S2 in the same position' (Trask 2000:336), one can conclude that the segments correspond *systematically*, and – as a result – are similar (or even identical in historical terms). When dealing with systematic *sound correspondences* in two or more language systems, one can further conclude that one is dealing with common descent. This conclusion is based on the fact that there are good reasons to assume that 'sound change is overwhelmingly regular, [and] we must expect a great degree of systematicity […] in the phonetic similarities between putatively related languages' (Hock & Joseph 2009:435). Lass (1997:130) calls this kind of resemblance 'genotypic' in opposition to a 'phenotypic' resemblance of phonetic segments which do not correspond systematically.

3.2. Sound correspondences and correspondence classes

The strict notion of diachronic similarity is prevalent in historical linguistics: Diachronic similarity is defined in absolute terms. Only if two segments are judged to correspond systematically, they are judged to be similar. In alignment analyses, which are the first stages of linguistic reconstruction, however, we need a heuristic which helps us to find *probably* corresponding segments rather than *absolutely* corresponding ones. Many authors (cf. e.g. Holzer 1996:174f, Szemerényi 1970:14f) emphasize that synchronic similarity can be neglected when establishing correspondence patterns, since, 'given a long enough time span,

[2]Cf. Hock (1991:557) for the Greek-Malay example.

almost any sound can change into any other sound' (Arlotto 1972:77). While it is true that there are good examples for sound changes which are difficult to explain on pure phonetic terms, most scholars, however, would probably also agree that sound change often *does* follow certain patters, that 'even the most divergent languages show examples of phonetic change which are remarkably similar' (Arlotto 1972:77), and that '[not] all changes are [...] equiprobable' (Lass 1997:136). The difference regarding the probability of certain sound changes to occur will also show up in the patterns of *sound correspondences* which can be observed in genetically related languages, whith certain patterns occurring more often and other ones being quite rare. Stating segment similarity in terms of *correspondence probability* will differ from a pure synchronic notion of similarity, yet it will, nevertheless, come closer to it than the strict notion of diachronic similarity, which is ignorant of phonetic realization.

The first one to attempt to derive a model of the probability of sound correspondences which is known to me was A. Dolgopolsky in a Russian paper from 1964 which appeared only 22 years later in an English translation (Dolgopolsky 1986). Based on a statistical analysis of the frequency of certain types of sound correspondences in a large (but unfortunately unpublished) etymological dataset of about 400 languages, he divided speech sounds into ten types (see Table 3), and 'thereby distinguished [them] in such a way that phonetic correspondences inside a "type" are more regular than those between different "types"' (Dolgopolsky 1986:35). Dolgopolsky's *sound classes* have been employed as a heuristic device for determining genetic language relationship in a few recent approaches (cf. Turchin et al. 2010, Mortarino 2009). In alignment analyses, however, sound classes have not been applied so far.

No.	**Class**	**Description**	**Example**
1	P	labial obstruents	p,b,f
2	T	dental obstruents	d,t,θ,ð
3	S	sibilants	s,z,ʃ,ʒ
4	K	velar obstruents, dental and alveolar affricates	k,g,ʦ,ʧ
5	M	labial nasal	m
6	N	remaining nasals	n,ɲ,ŋ
7	R	liquids	r,l
8	W	voiced labial fricative and initial rounded vowels	v,u
9	J	palatal approximant	j
10	ø	laryngeals and initial velar nasal	h,ɦ,ŋ

Table 3: Dolgopolsky's sound classes

4. A new method for multiple sequence alignment

The new method for multiple sequence alignment in historical linguistics which shall be presented in the following is based on the traditional framework of progressive multiple sequence alignment in evolutionary biology as it is presented in the CLUSTAL W package (Thompson et al. 1994). Apart from the outline given by this package, there are three

major modifications which have been made in order to suite the specific needs of alignment analyses in historical linguistics: (1) phonetic sequences are internally represented as *sound classes*; (2) methods of *position-specific scoring* are extended to cover *prosodic context*; and (3) alignments are automatically searched for *swapped sites*.

In the following, I shall briefly discuss these three basic modifications. Afterwards, the working procedure of the method shall be illustrated along with an example.

4.1. Sound classes as internal representation format

Offering a stochastically based intermediate solution between the two extreme positions of synchronic and diachronic similarity, sound classes seem especially suitable for automatic alignment analyses. Choosing strings of sound classes as internal representation format has many advantages: While alignment analyses in disciplines such as evolutionary biology always deal with the same *fixed set of characters*, such as the protein or DNA alphabets, the sound systems of the world's languages differ to a great degree (cf. the overview in Maddieson 2011), and the number of characters (including diacritics) of phonetic transcription systems such as the IPA will force the algorithm to handle a large bunch of phonetic values which will be meaningless in most applications. This is due to the fact that, on the one hand, there is a lot of variation regarding the way linguists transcribe languages: Apart from the difference between narrow and broad phonetic transcriptions, there are many cases in which linguists simply slightly differ in their judgments, especially in poorly studied languages. On the other hand, there are many correspondence patterns which occur so frequently, that it seems justified to give the respective sounds an identical value from the beginning. Thus, while probably no one would doubt that the velar unvoiced plosive [k] should be kept distinct from the labial unvoiced plosive [p], the distinction between the velar nasal [ŋ] and the uvular nasal [ɴ] is far less obvious and it doesn't seem likely that the performance of the algorithm will suffer if both sounds will be merged into one. In the current implementation of the method, the user can select between two different predefined sound class systems, a narrow one being based on the ASJP code (Brown et al. 2008) which reduces the full range of the IPA to 41 symbols, and a broad one being based on Dolgopolsky's original approach (Dolgopolsky 1986).

Apart from the sound class model, the scoring function which defines transition probabilities among sound classes is also of crucial importance. For Dolgopolsky's original sound class model (extended by a specific class for vowels), the scoring function employed by the method is pretty simple in so far as it penalizes all mismatches among consonant classes equally while forcing the program to avoid matches of the vowel class with one of the consonant classes. In contrast to this simple scoring scheme, the scoring function for the ASJP sound classes is far more complex. It is based on a substitution matrix of correspondence frequencies derived from a recent approach by Holman et al. (2011) in which an automated method for the detection of sound correspondences is applied to a large database containing word lists of more than 4000 languages (Wichmann et al. 2010) transcribed in ASJP code.

4.2. Position-specific scoring

In biological alignment algorithms it is common to treat specific positions of certain sequences differently by modifying the scores for the introduction of gaps (cf. Thompson et al. 1994). In the approach presented here, the idea of position-specific scoring is adopted to account for modified gap scores according to *prosodic context*. The main idea behind this modification is to account for the well-known fact that certain types of sound change are more likely to occur in specific prosodic contexts. For example, vowels are more likely to get lost than consonants, and consonants are more likely to get lost in syllable-final positions than in syllable-initial ones. It therefore seems fruitful to modify the gap scores in order to ease the introduction of gaps in prosodically weak positions while making it difficult to introduce them in prosodically strong positions. The algorithm employs a very simple method to account for this: In each progressive alignment step, a *prosodic profile* is constructed for all sequences. The prosodic profile is represented as a vector of numbers representing the relative sonority of all segments in a sonority hierarchy, going from lower numbers for less sonorous segments to higher numbers for more sonorous segments. The algorithm currently employs the sonority hierarchy given in Example 4, which follows Geisler (1992:30) with an additional sonority class for affricates.

(4)

plosives	*affricates*	*fricatives*	*nasals*	*liquids*	*glides*	*vowels*
1	2	3	4	5	6	7

Once the sonority profile of a sequence is calculated, all segments can be assigned to different prosodic contexts according to their position in the profile. Given the fact that syllabification is often language-specific (Hall 2000:226f), and that a given syllabification can change during the alignment process when more sequences are joined, the algorithm employs a modified strategy for the determination of prosodic context which is not based on syllabification. Given a sonority profile of a linguistic sequence, it is easy to determine peaks of *maximum* sonority, which will usually be vowels. It is further possible to determine whether a given segment (which does not appear in a maximum peak position) is in a position of *descending* or *ascending* sonority.

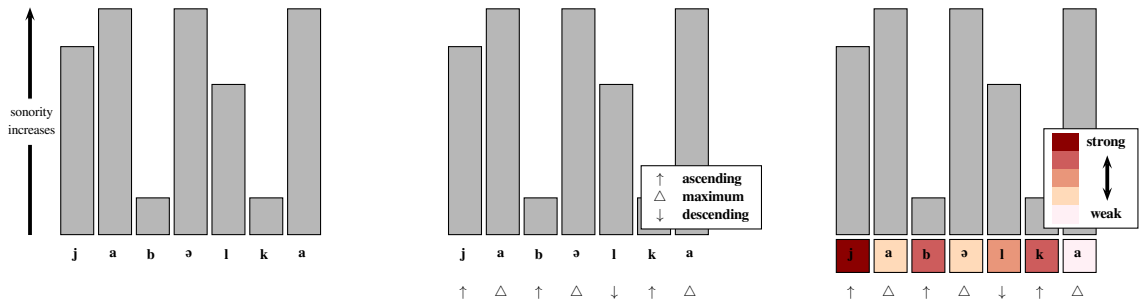

Figure 8: Relative weights derived from prosodic information

Apart from the positions of ascending, descending, and maximum sonority, the word-initial and the word-final positions are treated as separate specific environments, the

former being extremely strong and the latter being extremely weak in comparison to the other positions.[3] Once the positions are determined for a given sequence, the gap scores for each position can be modified according to its relative weight. Figure 8 illustrates, how the relative weights of all positions of Bulgarian *jabălko* [jabəlka] 'apple' are derived from the prosodic information: In a first step, only the sonority class is determined for each segment of the phonetic sequence. In the second step, positions of ascending, maximum, and descending sonority are calculated. In the final step, the relative weight of each position is determined.

4.3. Detection of swapped sites

Determining swapped sites (transpositions) in an alignment may be useful in some applications, since metathesis is a common sound change process which may even regularly occure in certain languages (Hock 1991:110). The method for the detection of swaps in a given multiple alignment is based on the idea that swaps are often aligned in a linear way in multiple alignment analyses, with one segment of a swapped site matching in two or more sequences and the remaining segments being matched with a gap to the left and the right of the match. This results in *complementary structures* in an MSA containing swapped sites (see Example 5). These structures can be easily detected. In order to guarantee that these structures really belong to swaps in an alignment, a scoring procedure for swaps in multiple alignments has to be applied.

(5)

Bulgarian	j	a	b	ə	l	-	k	a
Russian	j	a	b	-	l	ə	k	ə

Given the two hypothetical words `forma` and `froma`, we may align and score them in different ways. When allowing for swaps, the Damerau-Levenshtein distance metric (named after the work of Damerau 1964 and Levenshtein 1966) can be used. It penalizes swaps with 1, as illustrated in Example 6, yielding a total score of 1 for the two hypothetical sequences, since they only differ by one transposition.

(6)

f	o	r	m	a
f	r	o	m	a
0	1		0	0

Once we do not allow the algorithm to score swaps in this way, we arrive at a score of 2, if we adopt Levenshtein's traditional penalty system (Levenshtein 1966). Applying a different scoring system, where gaps are scored as 1, but mismatches are scored as 2, we can force the algorithm to avoid mismatches once complementary structures offer a better overall score, as shown in Example 7.

(7)

f	o	r	-	m	a
f	-	r	o	m	a
0	1	0	1	0	0

f	-	o	r	m	a
f	r	o	-	m	a
0	1	0	1	0	0

[3]Regarding the concept of 'strength' and 'weakness' in theories of sound change, cf. Geisler (1992).

This is exactly the situation we have in many MSAs, where sequences are linearily aligned, not allowing for crossed matches. Having detected complementary structures in an MSA, it is important to evaluate whether they really might point to swapped sites by scoring the alignment allowing for swaps. The problem with extensions of the dynamic programming algorithm for swaps is, however, that these are usually based on identity-conditions, where transposed characters are identical throughout an alignment, as the solution proposed by Wagner & Lowrance (1975). Oommen & Loke (1997) give a solution for an extension of the DPA which includes transpositions, but it applies only to pairwise alignments. For multiple sequence alignments it is important to have an alternative way to score swapped sites, where identity is given up in favour of a similarity or distance score which should be based on a sum-of-pairs scheme for MSAs.[4] This can be easily done by introducing a new character (+) into the sound class alphabet. The character scores only when it is matched with a gap, when matched with itself it scores 0, and it scores *infinite* when it is matched with any other character. The character is introduced into the gapped columns of a complementary site instead of the characters which are not a gap. The non-gap characters themselves are moved to the position where they would appear, if the site was not swapped. Transforming a given alignment in this way, the resulting score will be the same as in the Damerau-Levenshtein analysis, yielding 1 for two similar sequences which only differ by one transposition.

(8)

f	+	r	o	m	a		f	−	o	r	m	a
f	−	r	o	m	a		f	+	o	r	m	a
0	1	0	0	0	0		0	1	0	0	0	0

Given this scoring scheme, the method for swap detection works as follows: The algorithm first checks whether there are complementary structures in the alignment. If such structures are detected, the alignment is converted as shown in Example 8 and the total score of the alignment is calculated for the new alignment. If the new score exceeds the old one, the respective site is judged to be a swapped one.

Since alignment analyses are always a linear representation of sequences, it is not possible to display swapped sites properly. For the output one can choose between two formats, one merging the swapped sites into a single column and one which merges the three columns of a complementary site into two columns, as shown in Example 9.

(9)

f	or	m	a		f	o	r	m	a
f	ro	m	a		f	r	o	m	a

4.4. Working procedure

The working procedure of the method consists of seven stages as illustrated in Figure 9.[5] Every stage is accompanied by an example which shows the progress of the alignment of the four sequences Czech *jablko* [jabl̩ko], Bulgarian *jabălka* [jabəlka], Russian *jabloko*

[4]Regarding the calculation of sum-of-pairs scores, cf. Durbin et al. (2002:139f).

[5]The figure was inspired by the flow chart illustrating the CLUSTAL algorithm for MSA in evolutionary biology in Higgins & Sharp (1988:238).

[jabləkə], and Polish *jabłko* [japko] 'apple' at the respective stage. As mentioned before, the method closely follows the traditional way in which multiple sequences are progressively aligned in evolutionary biology. Comparing the linguistic working procedure with the biological one, there are four additional stages, namely Stage 1, Stage 5, Stage 6, and Stage 7.

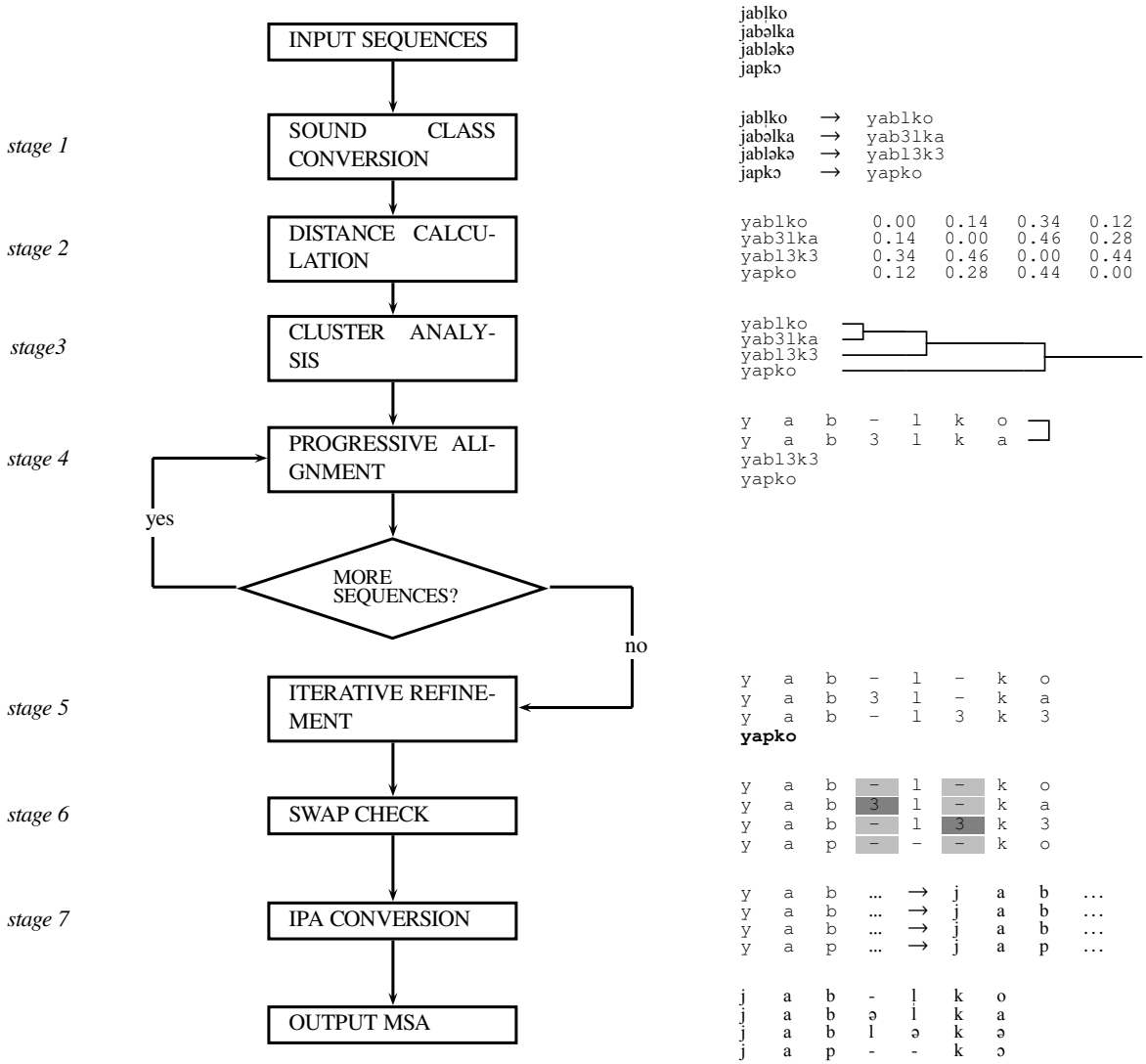

Figure 9: Working procedure of the basic algorithm

In Stage 1, the input sequences which are given in IPA transcription are converted to sound classes, e.g. the sequence Czech [jabl̩ko] is converted to `yablko` according to the ASJP sound class model. In Stage 2, a distance matrix is computed from the pairwise similarity scores of all sound class sequences. The approach of Downey et al. (2008) is used for the conversion of similarity into distance scores. In Stage 3, the sound class sequences are clustered by applying a simple agglomerative hierarchical cluster algorithm to the

distance matrix calculated in Stage 2.[6] The clustering procedure yields the guide tree which is important for the progressive alignment procedure applied in Stage 4. Stage 4 is the core of the alignment process. In this stage, all sequences are stepwise aligned with each other, following the branching order of the guide tree (cf. the description in Section 2.2). In Stage 5, iterative refinement methods are applied to the already aligned sequences in order to account for possible errors resulting from misaligned sequences.[7] The method for swap detection which was described above, is applied in Stage 6. As can be seen from the example, the algorithm identifies a swap for the alignment of the four input sequences. In Stage 7, all sequences are converted back from their internal representation format to their original IPA format.

4.5. Implementation of the method

The method is implemented as part of the LingPy library (List 2011). LingPy is a suite of open source Python modules for sequence comparison, distance analyses, data operations and visualization methods in quantitative historical linguistics.[8]

5. Performance of the method

In order to test the performance of alignment algorithms it is common to compare their output with independently compiled gold standard datasets which serve as a benchmark. By comparing the *reference alignment* with the *test alignment* produced by the algorithm, an assessment of the accuracy of the algorithm can be given.

For this study, a benchmark dataset originally compiled for the study of Prokić et al. (2009) was used. In this pilot study on multiple alignment of phonetic sequences, the authors employed the ALPHAMALIG algorithm to a dataset of phonetic transcriptions of Bulgarian dialects (taken from the Buldialect project[9]). In contrast to the new method presented here, ALPHAMALIG was not especially designed for the alignment of phonetic sequences.[10] The gold standard used in the study (Prokić et al. 2009) was kindly provided by the authors and covers 152 manually edited multiple sequence alignments for a total of 192 taxa (dialect points). Before reporting the results of the comparison of the performance of the method presented in this paper with that of the ALPHAMALIG algorithm, I will briefly point to some general issues regarding the evaluation of alignment analyses based on benchmark datasets.

---

[6]For the details of agglomerative hierarchical clustering, see e.g. Tan et al. (2006:515-526).

[7]To go into the details of this step would go beyond the scope of this paper. For a general account on iterative procedures in MSA, cf. Durbin et al. (2002:148f).

[8]See http://lingulist.de/lingpy/.

[9]See http://www.sfs.uni-tuebingen.de/dialectometry/ for the Buldialect project, and see http://alggen.lsi.upc.es/recerca/align/alphamalig/intro-alphamalig.html for the ALPHAMALIG algorithm.

[10]Cf. Prokić et al. (2009) for more details regarding the ALPHAMALIG algorithm.

5.1. Evaluation

The comparison of alignments produced by different methods with a benchmark dataset is a complicated task, and all methods, which have been proposed so far, be it in biological or linguistic applications, bear certain shortcomings. Generally, when comparing alignments, there are two perspectives which can be taken: (1) The *column perspective*, which focuses on the columns in both the reference and the test alignment; and (2) the *row perspective*, which focuses on the rows in both the reference and the test alignment. The simplest way to compare a reference with a test alignment is to base the comparison solely on one of the two perspectives, and calculate either the *percentage of identical columns* (PIC), or the *percentage of identical rows* (PIR).

Both measures, however, are problematic, as can easily be demonstrated: Table 4 gives five examplary cases on how alignments can differ from each other. The left alignment is an exemplary reference alignment, and the two alignments on the right are one and the same possible deviation from the reference alignment where the differences in terms of columns and rows are shaded in lightgray. As can be seen from the examples, it may be difficult to say which of a couple of test alignments is closest to a reference alignment, when relying on either of both scores: When relying on the PIR score, errors resulting from identical erroneous decisions cannot be properly distinguished from errors resulting from different erroneous decisions as in Examples 1 and 2 of Table 4. Furthermore, certain errors will be exaggerated, as in Examples 3 and 4 of Table 4 where no rows are identical although the alignments only differ in two columns. Relying solely on the PIC score, however, a single erroneously aligned sequence may likewise greatly exaggerate the score (see Example 5 of Table 4).

	Seqs	Reference	Test (column)	Test (row)
1	Seq1	A – C D E	A – C D E	A – C D E
	Seq2	A – C – E	A C – – E	A C – – E
	Seq3	A – C D E	A C – D E	A C – D E
	Seq4	A B – D E	A B – D E	A B – D E
	Seq5	A – – D –	A – – D –	A – – D –
2	Seq1	A – C D E	A – C D E	A – C D E
	Seq2	A – C – E	A – – C E	A – – C E
	Seq3	A – C D E	A C – D E	A C – D E
	Seq4	A B – D E	A B – D E	A B – D E
	Seq5	A – – D –	A – – D –	A – – D –
3	Seq1	A – C D E	A – C D – E	A – C D – E
	Seq2	A – C – E	A C – – E –	A C – – E –
	Seq3	A – C D E	A C – D E –	A C – D E –
	Seq4	A B – D E	A B – D E –	A B – D E –
	Seq5	A – – D –	A – – D – –	A – – D – –
4	Seq1	A – C D E	A C D E	A C D E
	Seq2	A – C – E	A C – E	A C – E
	Seq3	A – C D E	A C D E	A C D E
	Seq4	A B – D E	A B D E	A B D E
	Seq5	A – – D –	A – D –	A – D –
5	Seq1	A – C D E	A – C D E	A – C D E
	Seq2	A – C – E	A – C – E	A – C – E
	Seq3	A – C D E	A – C D E	A – C D E
	Seq4	A B – D E	A B – D E	A B – D E
	Seq5	A – – D –	– A D – –	– A D – –

Table 4: Differences between reference and test alignments

A common evaluation measure for alignment accuracy in evolutionary biology is based on the sum-of-pairs score (SP, cf. Thompson et al. 1999). This score is defined as the size of the intersection of aligned pairs of residues in reference and test alignment divided by the size of aligned pairs of residues in the reference alignment. Despite the fact that

the SP score is a very common evaluation measure in evolutionary biology (Aniba et al. 2010), its suitability as an evaluation measure may be questioned: Since the SP score takes only identical pairs in both alignments into account, ignoring non-identical pairs in the test alignment, the measure fails to discriminate between differences in reference and test alignment as those marked in Case 4 of Table 4, where the SP score is 1, indicating identity, despite the fact that both alignments are different. Instead of the SP score, it seems more adequate to compute the Jaccard coefficient (JC) which is defined as the size of the intersection of two sets divided by the size of their union (Batagelj & Bren 1995:80).

Prokić et al. (2009) propose two enhanced methods to evaluate the performance of their alignment algorithm on the gold standard of Bulgarian dialects. The first score, the *column dependent evaluation* (CDE) is an enhancement over the simple PIC where the similarity of the columns is evaluated instead of their identity (see Prokić et al. 2009:20f for details). The second score is based on the *modified rand index* (MRI). While the CDE score is sensitive to order, the 'MRI itself only takes into account the quality of each column separately since it simply checks whether the same elements are together in the candidate alignment as in the gold-standard alignment' (Prokić et al. 2009:21).

	PIC	**PIR**	**CDE**	**MRI**	**SP**	**JC**
1	0.60	0.60	0.84	0.90	0.89	0.80
2	0.40	0.60	0.84	0.81	0.83	0.65
3	0.80	0.00	0.80	0.92	0.83	0.83
4	0.60	0.00	0.27	0.93	1.00	0.86
5	0.20	0.80	0.84	0.71	0.72	0.59

Table 5: Comparison of evaluation scores

In Table 5, all scores for the evaluation measures applied to the five examples in Table 4 are given. As can be seen from the table, the JC and MRI scores properly distinguish all differences between reference and test alignments.

5.2. Results

Based on the new method for progressive MSA, the 152 alignments of the benchmark dataset of Bulgarian dialects were analyzed using the two standard sound class models described in Section 4.1, i.e. the Dolgopolsky model (LinPy-DOLGO), and the ASJP model (LingPy-ASJP).[11]

	Perf. Alm.	**PIC**	**PIR**	**CDE**	**MRI**	**SP**	**JC**
LingPy-DOLGO	123 (81 %)	0.8704	0.8825	0.9494	0.9844	0.9850	0.9742
LingPy-ASJP	132 (87 %)	0.9313	0.9531	0.9763	0.9902	0.9901	0.9834
ALPHAMALIG	103 (68 %)	0.8401	0.7632	0.9324	0.9824	0.9825	0.9743

Table 6: Results of the evaluation

[11]The data for all analyzes, including the concise results for the evaluation is avalailable under http://lingulist.de/lingpy/sole-data.zip.

Table 6 shows the results for the evaluation of the LingPy method for the two models in comparison with the results achieved by the ALPHAMALIG algorithm. As the table shows, both models of the LingPy algorithm perform better than ALPHAMALIG throughout all evaluation scores, the differences being especially evident when comparing the simple PIR scores, where LingPy-ASJP aligns 95% of all sequences in the same way as the gold standard, while ALPHAMALIG gets only 76% of identical rows.

As a closer analysis of the output of the different methods compared with the gold standard shows, one main advantage of the new method lies in its ability to detect swapped sites: Of the 21 test alignments in the database which contain swaps, LingPy-ASJP correctly identifies 19, and 13 of them are further aligned without errors. Figure 10 gives an example for a correct alignment of a swapped site produced by LingPy-ASJP in contrast to an erroneous alignment produced by ALPHAMALIG in Figure 11 (characters are colored according to the Dolgopolsky sound class scheme).

Aldomirovci	v	r	ɑ	-	ʧ	ɑ	m
Asparuhovo	v	r	ɤ	ʃ	t	ɑ	m
Panagjurishte	v	ɤ	r	ʃ	t	ə	m
Rakovica	v	ɾ	-	ʃ	t	ɑ	m
Stambolovo	v	r	ɤ	ɕ	t	ə	m

Figure 10: MSA 21: LingPy-ASJP

Aldomirovci	v	-	r	ɑ	-	ʧ	ɑ	m
Asparuhovo	v	-	r	ɤ	ʃ	t	ɑ	m
Panagjurishte	v	ɤ	r	-	ʃ	t	ə	m
Rakovica	v	-	ɾ	-	ʃ	t	ɑ	m
Stambolovo	v	-	r	ɤ	ɕ	t	ə	m

Figure 11: MSA 21: ALPHAMALIG

The differences between LingPy-ASJP and LingPy-DOLGO further illustrate, that the automatic detection of swapped sites is not the sole advantage of the algorithm, but that the model of sound classes and the scoring function are also of crucial importance for its performance. While the DOLGO model clusters sounds in a very broad way and employs a very rough scoring function, the ASJP model is based on a more subtle sound class model and a scoring function which was derived from a large empirical basis. The differences between the two models are illustrated in Figures 12 and 13: While the DOLGO model always prefers consonant-consonant matches over matches of consonants and vowels, no matter whether the consonant classes are similar or not, the ASJP model yields an alignment in accordance with the gold standard, where the preference over the avoidance of vowel-consonant matches is given to the matching of the phonetically more similar characters.

Aldomirovci	u	n	e	t	r	e
Asparuhovo	v	-	ɤ	t	rʲ	ə
Babjak	f	n	e	t	r	e
Bachkovo	v	-	ɑ	t	rʲ	ə
Bagrenci	u	n	e	t	r	e

Figure 12: MSA 27: LingPy-ASJP

Aldomirovci	u	n	e	t	r	e
Asparuhovo	-	v	ɤ	t	rʲ	ə
Babjak	f	n	e	t	r	e
Bachkovo	-	v	ɑ	t	rʲ	ə
Bagrenci	u	n	e	t	r	e

Figure 13: MSA 27: LingPy-DOLGO

6. Conclusion

While multiple sequence alignment has a long tradition in evolutionary biology, it is still in its infancy in historical linguistics and dialectology. The algorithm described in this paper only slightly differs from the common framework of progressive multiple sequence alignment in biology which was developed during the last twenty years. The main improvements of the algorithm lie in the incorporation of some specifically linguistic features, such as the use of sound classes along with specific scoring functions, or the modification of gap scores based on prosodic profiles.

When implementing the new method I intentionally took care to keep it flexible regarding most of its parameters. Using the LingPy library, it is easy to create new sound class models or to modify given ones in dependence of the data one wishes to analyze. In my opinion, this flexibility is important when trying to get multiple sequence alignment in historical linguistics out of its infancy: Despite more than 200 years of research on genetic language relationship and sound change, we still do not fully understand the phenomena in all their complexity, and all work in quantitative historical linguistics remains provisional until we drastically increase our data basis and start to get theory and practice closer to each other.

Acknowledgements

I am deeply indebted to Jelena Prokić, Martijn Wieling, and John Nerbonne for providing me with the gold standard dataset of Bulgarian dialects, the source code for the calculation of the CDE and MRI scores, and the results of their analysis using the ALPHAMALIG algorithm. I would also like to thank Hans Geisler for helpful and inspiring critics.

Johann-Mattis List
Heinrich Heine University Düsseldorf
listm@phil.uni-duesseldorf.de

References

Aniba, M. R., O. Poch & J. D. Thompson (2010). Issues in bioinformatics benchmarking: the case study of multiple sequence alignment. *Nucleic Acids Research* 38:21, p. 7353–7363.

Anttila, R. (1972). *An introduction to historical and comparative linguistics*. Macmillan, New York.

Arlotto, A. (1972). *Introduction to historical linguistics*. Mifflin, Boston.

Batagelj, V. & M. Bren (1995). Comparing resemblence measures. *Journal of Classification* 12, pp. 73–90.

Böckenbauer, H.-J. & D. Bongartz (2003). *Algorithmische Grundlagen der Bioinformatik*. Teubner, Stuttgart and Leipzig and Wiesbaden.

Brown, C. H., E. W. Holman, S. Wichmann, V. Velupillai & M. Cysouw (2008). Automated classification of the world's languages. *Sprachtypologie und Universalienforschung* 61:4, pp. 285–308.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the Association for Computing Machinery* 7:3, pp. 171–176.

Dolgopolsky, A. B. (1986). A probabilistic hypothesis concerning the oldest relationships among the language families of northern Eurasia. Shevoroshkin, V. V. (ed.), *Typology, Relationship and Time*, Karoma Publisher, Ann Arbor, pp. 27–50.

Downey, S. S., B. Hallmark, M. P. Cox, P. Norquest & S. Lansing (2008). Computational feature-sensitive reconstruction of language relationships: Developing the ALINE distance for comparative historical linguistic reconstruction. *Journal of Quantitative Linguistics* 15:4, p. 340–369.

Durbin, R., S. R. Eddy, A. Krogh & G. Mitchinson (2002). *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, 7 edn.

Feng, D. F. & R. F. Doolittle (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution* 25:4, pp. 351–360.

Geisler, H. (1992). *Akzent und Lautwandel in der Romania*. No. 38 in Romanica Monacensia, Narr, Tübingen.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences*. Cambridge University Press, Cambridge.

Hall, T. A. (2000). *Phonologie. Eine Einführung*. de Gruyter, Berlin and New York.

Higgins, D. G. & P. M. Sharp (1988). Clustal. *Gene* 73, pp. 237–244.

Hock, H. H. (1991). *Principles of historical linguistics*. Mouton de Gruyter, Berlin, 2 edn.

Hock, H. H. & B. D. Joseph (2009). *Language history, language change and language relationship*. Mouton de Gruyter, Berlin and New York, 2 edn.

Holman, E. W., C. H. Brown & S. Wichmann (2011). Sound correspondences in the world's languages. Online document, URL http://wwwstaff.eva.mpg.de/~wichmann/wwcPaper23.pdf.

Holzer, G. (1996). *Das Erschließen unbelegter Sprachen*. No. 1 in Schriften über Sprachen und Texte, Lang, Frankfurt am Main.

Kommission "Selbstkontrolle in der Wissenschaft" der Universität Bayreuth (2011). Bericht an die Hochschulleitung der Universität Bayreuth aus Anlass der Untersuchung des Verdachts wissenschaftlichen Fehlverhaltens von Herrn Karl-Theodor Freiherr zu Guttenberg. Online ressource, URL http://www.uni-bayreuth.de/presse/info/2011/Bericht_der_Kommission_m__Anlagen_10_5_2011_.pdf.

Lass, R. (1997). *Historical linguistics and language change*. Cambridge University Press, Cambridge.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10:8, pp. 707–710.

List, J.-M. (2011). Lingpy. Online ressource, URL http://lingulist.de/lingpy/.

Maddieson, I. (2011). Consonant inventories. Dryer, M. & M. Haspelmath (eds.), *The world atlas of language structures online*, Max Planck Digital Library, Munich, URL http://wals.info/chapter/1.

Mortarino, C. (2009). An improved statistical test for historical linguistics. *Statistical Methods and Applications* 18:2, pp. 193–204.

Needleman, S. B. & C. D. Wunsch (1970). A gene method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48, pp. 443–453.

Oommen, B. J. & R. K. S. Loke (1997). Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. *Pattern Recognition* 30:5, pp. 789–800.

Prokić, J., M. Wieling & J. Nerbonne (2009). Multiple sequence alignments in linguistics. *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*, Association for Computational Linguistics, Stroudsburg, PA, pp. 18–25.

Saitou, N. & M. Nei (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* 4:4, pp. 406–425.

Sokal, R. R. & C. D. Michener (1958). A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin* 28, pp. 1409–1438.

Szemerényi, O. (1970). *Einführung in die vergleichende Sprachwissenschaft*. Wissenschaftliche Buchgesellschaft, Darmstadt.

Tan, P.-N., M. Steinbach & V. Kumar (2006). *Introduction to data mining*. Pearson, Boston.

Thompson, J. D., D. G. Higgins & T. J. Gibson (1994). CLUSTAL W. *Nucleic Acids Research* 22:22, p. 4673–4680.

Thompson, J. D., F. Plewniak & O. Poch (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research* 27:13, pp. 2682–2690.

Trask, R. L. (ed.) (2000). *The dictionary of historical and comparative linguistics*. Edinburgh University Press, Edinburgh.

Turchin, P., I. Peiros & M. Gell-Mann (2010). Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship* 3, pp. 117–126.

Wagner, R. A. & M. J. Fischer (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery* 21:1, pp. 168–173.

Wagner, R. A. & R. Lowrance (1975). An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery* 22:2, pp. 177–183.

Wichmann, S., A. Müller, V. Velupillai, C. H. Brown, E. W. Holman, P. Brown, S. Sauppe, O. Belyaev, M. Urban, Z. Molochieva, A. Wett, D. Bakker, J.-M. List, D. Egorov, R. Mailhammer, D. Beck & H. Geyer (2010). The ASJP database (version 13). Online document, URL http://email.eva.mpg.de/~wichmann/listss13.zip.