Distanz- und Alignmentanalysen in der historischen Linguistik

Johann-Mattis List

22. Januar 2010

Im Gegensatz zu quantitativen Methoden, wie der Lexikostatistik (vgl. Swadesh 1950,1952 & 1955), der Etymostatistik (vgl. Starostin 2000, Shapiro 2007) oder der Separation Base Method (vgl. Holm 2000 & 2007), die auf dem Vergleich von Mengen, also ungeordneten Elementen, beruhen, konzentrieren sich Sequenzanalysen auf den Vergleich von geordneten Listen von Objekten. In diesem Zusammenhang soll ein kurzer Überblick über die beiden gängigsten Methoden automatischer Sequenzvergleiche in der historischen Linguistik gegeben werden: Distanz- und Alignmentanalysen. Vorgestellt werden sollen dabei der grundlegende dynamische Programmieralgorithmus, der bei Sequenzvergleichen in der Bioinformatik, der Linguistik und der Informatik Verwendung findet (vgl. Wagner & Fischer 1974), und seine Implementierung und Modifizierung für Alignment- und Distanzanalysen in der historischen Linguistik (vgl. u.a. Kondrak 2002, Holman *et al.* 2008b,Covington 1996). Darüber hinaus sollen grundlegende Probleme der derzeit gängigsten Ansätze angesprochen und diskutiert werden.

1 Einige Grundlegende Anmerkungen

1.1 Mengen- vs. Sequenzvergleiche

Im Folgenden wird eine grundlegende Unterscheidung zwischen Mengen- und Sequenzvergleichen vorgenommen und zur Diskussion gestellt: Während Mengenvergleiche in dieser Darstellung auf dem Vergleich einer gewissen Anzahl ungeordneter, distinkter Elemente (Wörter im Basislexikon, Buchstaben in Wörtern) beruhen, ist für Sequenzvergleiche die Anordnung der Elemente entscheidend, da Distinktivität der Elemente erst durch diese möglich wird. Wichtig in diesem Zusammenhang ist insbesondere, dass auf Sequenzvergleichen beruhende Distanzanalysen eine Alinierung der Sequenzen voraussetzen, d.h. die Elemente, die einander in verschiedenen Sequenzen entsprechen, die Korrespondenzen, müssen ermittelt werden, um die Distanz zwischen den Sequenzen bestimmen zu können.[1]

[1] Kruskal (1983:201f) trifft eine Entscheidung zwischen Sequenzvergleichen, bei denen die Korrespondenzen bekannt sind und solchen, bei denen die Korrespondenzen nicht bekannt sind. Die folgende Darstellung beschäftigt sich nur mit Korrespondenzvergleichen der letztgenannten Art.

Monogrammatische Segmentierung	θ	i	ɣ	a	t	ɛ	r	a	
Bigrammatische Segmentierung	-θ	θi	iɣ	ɣa	at	tɛ	ɛr	ra	a-
Trigrammatische Segmentierung	--θ	-θi	θiɣ	iɣa	ɣat	atɛ	tɛr	ɛr-	r--

Tabelle 1: Mono- bi und trigram-basierte Segmentierung

1.2 Unigrammatische und *n*-grammatische Sequenzvergleiche

Bei Sequenzvergleichen kann grundlegend zwischen unigrammatischen und *n*-grammatischen Sequenzvergleichen unterschieden werden (vgl. Kondrak 2005). Während unigrammatische Sequenzvergleiche eine Sequenz in Segmente der Länge 1 (*unigram*) zerlegen, liegt *n*-grammatischen Elementen eine Sequenziering in Segmente der Länge *n* zugrunde (vgl. Tabelle 1, für eine Diskussion und Anwendungsbeispiele *n*-grammatischer Sequenzvergleiche vgl. Heeringa *et al.* 2006). In dieser Darstellung werden aus Gründen des Umfangs und des eigenen Arbeitsstandes nur unigrammatische Sequenzvergleiche berücksichtigt.

1.3 Paarweise vs. multiple Alinierung

Bei der Distanz- und Alinierungsanalyse von Sequenzen muss unterschieden werden zwischen paarweiser und multipler Alinierung: Während paarweise Alinierung sich auf die Alinierung von zwei Sequenzen beschränkt, wird in der multiplen Alinierung nach der optimalen Alinierung von mehr als zwei Sequenzen gesucht. Die scharfe Trennung von multipler und paarweiser Alinierung ist darauf zurückzuführen, dass die Laufzeit der zur Verfügung stehenden Algorithmen mit der Anzahl der untersuchten Sequenzen exponentiell ansteigt (vgl. Morgenstern *et al.* 1996, Bilu *et al.* 2006, 409). Aus diesem Grunde werden in der Biologie entweder komplizierte Heuristiken eingesetzt, oder die zu alinierenden Sequenzen werden zunächst paarweise aliniert, um dann in einem weiteren Schritt mit Hilfe von Clusterverfahren die optimale Alinierung von mehreren Sequenzen zu ermitteln.

d	ɔː	-	-	t^h	ɚ	-	-
θ	i	ɣ	a	t	ɛ	r	a
t^h	ɔ	x	-	t^h	ɐ	-	-

Tabelle 2: Multiple Alinierung von Sequenzen

Alle bisher in der Linguistik vorgenommenen Alinierungsmethoden beschänken sich meines Wissens nach auf die paarweise Alinierung von Sequenzen. Das Problem der multiplen Alinierung wurde bisher noch nie explizit in Angriff genommen, was zu bedauern ist, da zu erwarten ist, dass multiple Alinierung dem auf kumulativer Evidenz (Makaev 1977, 88) beruhenden Vorgehen der historischen Linguistik näher kommt. Aliniert man bspw. die Wörter gr. [θigatɛra] 'Tochter' und engl. [dɔːt^{h}ɚ] 'Tochter', ohne gleichzeitige Kenntnis der dt. Form [t^{h}ɔxt^{h}ɐ] 'Tochter', so liefern Algorithmen wie der von Covington (1996) oder Kondrak (2002) falsche Alinierungen für die englisch-griechische Paarung: aufgrund des fehlenden Velars im Englischen wird engl. [d] gr. [g] gegenübergestellt. Die Alinierung von engl. [dɔːt^{h}ɚ] und dt. [t^{h}ɔxt^{h}ɐ] bzw. von dt. [t^{h}ɔxt^{h}ɐ] und gr. [θigatɛra] wird von beiden Algorithmen richtig vorgenommen, und würde daher auch in einem multiplen Alignment zu dem richtigen Ergebnis führen, da schlicht und

Entscheidung	**Bedingung**	**Kosten**
Gegenüberstellung	Identität von Segmenten	0
	Verschiedenheit von Segmenten	1
Einfügen und Ersetzen		1

Tabelle 3: Die Vergleichsfunktion zur Berechnung der Levenshtein-Distanz

einfach mehr Evidenz als in einer paarweisen Alinierung in eine derartige Alinierung eingebracht werden kann (vgl. Tabelle 2). Da, wie bereits erwähnt, multiple Alinierungen in der historischen Linguistik bisher noch nicht praktiziert wurden, werde ich mich in der folgenden Darstellung auf paarweise Alinierungen beschränken.

1.4 Der dynamische Programmieralgorithmus

Nahezu alle Ansätze zu computergestützten Analyse von Sequenzdistanzen, sei es in der historischen Linguistik, der Biologie oder der Informatik, gründen auf dem im Jahre 1974 von Wagner & Fischer beschriebenen dynamischen Programmieralgorithmus, der eine Computerimplementierung der Levenshtein-Distanz (vgl. Levenshtein 1966) darstellt [2] Da dieser Algorithmus für alle Alignmentalgorithmen und für alle auf Alinierung beruhenden Algorithmen zur Distanz-Berechnung von großer Bedeutung ist, soll er im Folgenden detailliert vorgestellt werden.

Die Grundidee des Wagner-Fischer-Algorithmus besteht im Erstellen einer Matrix mit allen möglichen Segmententsprechungen zweier Sequenzen (vgl. Tabelle 4). In einem weiteren Schritt wird nun sukzessive nach dem kürzesten Weg vom linken oberen Rand hin zum rechten unteren Rand der Matrix gesucht, indem kumulativ die Kosten für jeden möglichen Weg berechnet werden. Die Kosten werden durch eine spezielle Funktion berechnet (zuweilen 'Bewertungsfunktion' genannt, ich werde im Folgenden von 'Vergleichsfunktion' sprechen), indem zunächst die jeweiligen Segmente miteinander verglichen werden und aus einer Menge verschiedener Kosten die jeweils günstigste Möglichkeit ausgesucht wird. Dabei wird zwischen zwei grundlegenden Möglichkeiten unterschieden: dem Gegenüberstellen von Segmenten[3] und dem Ersetzen bzw. Einfügen von Segmenten (*indel*: *insertion* und *deletion*).

Bei der Berechnung der traditionellen Levenshtein-Distanz sieht die Vergleichsfunktion beispielsweise wie in Tabelle 3 beschrieben aus, die Funktionen für das Aufsummieren der Kosten können jedoch weitaus komplexer sein. Tabelle 5 zeigt exemplarisch die jeweiligen Entscheidungen, die durch die Anwendung der reinen Levenshtein-Funktion für die graphemische Repräsentation der Wörter 'daughter' und 'tochter' gefällt wurden, Tabelle 6 zeigt die numerischen Werte, die sich bei dieser Berechnung aufsummieren.

Da die Werte aufsummiert werden, ist die Distanz zwischen zwei Segmenten stets in der untersten rechten Spalte der Matrix zu finden. Während die Distanz stets den kürzesten Weg, der auf Grundlage der Vergleichsfunktion zurückgelegt wurde, repräsentiert, stellt die Alinierung der Sequenzen, wie sie bspw. in Tabelle 4 dargestellt ist, nicht immer die einzige Möglichkeit dar. Eine Distanz kann in vielen Fällen durch mehrere Alinierungen dargestellt werden, die aus Perspektive der auf der Vergleichsfunktion gewonnenen Distanz gleichwertig sind.

[2] Bereits im Jahre 1970 veröffentlichten Needleman & Wunsch (1970) einen Algorithmus zur Alinierung von Proteinsequenzen, welcher dem von Wagner & Fischer (1974) beschriebenen sehr ähnelt, jedoch anstelle von Distanzen von Ähnlichkei-

-	-	-	t	-	o	-	c	-	h	-	t	-	e	-	r
d	-	d	t	d	o	d	c	d	h	d	t	d	e	d	r
a	-	a	t	a	o	a	c	a	h	a	t	a	e	a	r
u	-	u	t	u	o	u	c	u	h	u	t	u	e	u	r
g	-	g	t	g	o	g	c	g	h	g	t	g	e	g	r
h	-	h	t	h	o	h	c	h	h	h	t	-	e	h	r
t	-	t	t	t	o	t	c	t	h	t	t	h	e	t	r
e	-	e	t	e	o	e	c	e	h	e	t	e	e	e	r
r	-	r	t	r	o	r	c	r	h	r	t	r	e	r	r

Tabelle 4: Vergleichsmatrix für den Wagner-Fischer-Algorithmus

-	-	-	t	-	o	-	c	-	h	-	t	-	e	-	r
d	-	d	t	-	o	-	c	-	h	-	t	-	e	-	r
a	-	a	-	a	o	-	c	-	h	-	t	-	e	-	r
u	-	u	-	u	-	u	c	-	h	-	t	-	e	-	r
g	-	g	-	g	-	g	-	g	h	-	t	-	e	-	r
h	-	h	-	h	-	h	-	h	h	-	t	-	e	-	r
t	-	t	t	t	-	t	-	t	-	t	t	-	e	-	r
e	-	e	-	e	o	e	-	e	-	e	-	e	e	-	r
r	-	r	-	r	-	r	c	r	-	r	-	r	-	r	r

Tabelle 5: Vergleichsmatrix nach der Bearbeitung

-	-	0	-	t	1	-	o	2	-	c	3	-	h	4	-	t	5	-	e	6	-	r	7
d	-	1	d	t	1	-	o	2	-	c	3	-	h	4	-	t	5	-	e	6	-	r	7
a	-	2	a	-	2	a	o	2	-	c	3	-	h	4	-	t	5	-	e	6	-	r	7
u	-	3	u	-	3	u	-	3	u	c	3	-	h	4	-	t	5	-	e	6	-	r	7
g	-	4	g	-	4	g	-	4	g	-	4	g	h	4	-	t	5	-	e	6	-	r	7
h	-	5	h	-	5	h	-	5	h	-	5	h	h	4	-	t	5	-	e	6	-	r	7
t	-	6	t	t	5	t	-	6	t	-	6	t	-	5	t	t	4	-	e	5	-	r	6
e	-	7	e	-	6	e	o	6	e	-	7	e	-	6	e	-	5	e	e	4	-	r	5
r	-	8	r	-	7	r	-	7	r	c	7	r	-	7	r	-	6	r	-	5	r	r	4

Tabelle 6: Vergleichsmatrix mitsamt den numerischen Werten

ten zwischen Segmenten ausgeht. Wem die Ehre für die Entdeckung des dynamischen Programmieralgorithmus in diesem Zusammenhang tatsächlich gebührt, ist für mich zu diesem Zeitpunkt schwer festzustellen, da in unterschiedlichen Wissenschaftszweigen diesbezüglich auffällig unterschiedliche Zitationstraditionen vorliegen.

[3] Dieses wird meist als Ersetzung (*substitution*) bezeichnet, was allerdings nicht für alle Distanzen gelten muss.

2 Möglichkeiten der Erweiterung und Verbesserung des Wagner-Fischer-Algorithmus

Zur Verbesserung und Erweiterung des Wagner-Fischer-Algorithmus, oder besser gesagt, zur Anpassung des Algorithmus an die Bedürfnisse der historischen Sprachwissenschaft oder anderer Einsatzgebiete, kommen verschiedene Möglichkeiten in Betracht. Unterschieden werden kann hier zwischen Erweiterungen, die in den Algorithmus selbst eingreifen, Erweiterungen, welche die ursprüngliche Darstellung der Segmente betreffen und Erweiterungen, die die Vergleichsfunktion betreffen.

2.1 Erweiterungen des Algorithmus

Als Beispiele für Erweiterungen des Algorithmus seien im Folgenden genannt: die Einbeziehung von Transpositionen, die geringere Bestrafung einer Reihe konsekutiver im Gegensatz zu singulären *indels*, das Hinzufügen von Kompressionen bzw. Expansionen als Möglichkeit der Alinierung und die Durchführung lokaler Alinierungen, die sich auf Substrings beziehen, im Gegensatz zu globalen Alinierungen.

Transpositionen

Transpositionen (Metathesen) spielen insbesondere bei der automatischen Fehlerkorrektur eine große Rolle, weil sie dort besonders häufig auftreten, und wurden auch zu diesem Zweck früh explizit in den Sequenzvergleich miteinbezogen (vgl. Damerau 1964). Die Integration in den dynamischen Programmieralgorithmus von Wagner & Fischer (1974) wurde erstmals in Wagner & Lowrance (1975) beschrieben. Die bekannteste Anwendung stellt die Damerau-Levenshtein-Distanz dar, welche Transpositionen mit 1 bestraft.

Während die meisten Realisierungen des Algorithmus, die mir bekannt sind, diesen um eine Extra-Zeile ergänzen, die nach der Auswertung der Kosten zusätzlich nach möglichen Transpositionen sucht, scheint es mir sinnvoller, die Transpositionsoperation direkt in die Auswertung der Kosten zu integrieren, und statt einzelnen Segmenten jeweils auch doppelte Segmente einander gegenüberzustellen, welche umgekehrt angeordnet werden, bei der Alinierung werden diese Segmente dann jeweils zusammen ausgegeben. Dadurch wird auch deutlich, dass vom Algorithmus eine Transposition angesetzt wurde. Tabelle 7 gibt ein Beispiel für die unterschiedliche Alinierung der Wörter it. *formaggio* 'Käse' und fr. *fromage* 'Käse'.

Levenshtein	f	r	o	m	a	g	e	-	-
	f	o	r	m	a	g	g	i	o
Damerau-Levenshtein	f	ro		m	a	g	e	-	-
	f	or		m	a	g	g	i	o

Tabelle 7: Levenshtein und Damerau-Levenshtein

Angesichts der Tatsache, dass Metathesen in der historischen Linguistik zwar bekannt sind, jedoch recht selten vorkommen, bzw. viel komplexere Ergebnisse produzieren, bei denen nicht nur einzelne Laute vertauscht werden, stimme ich mit Kondrak (2002:50) überein, dass es sich nicht lohnt, diese in einen Algorithmus zu integrieren, der für Sequenzvergleiche in der historischen Linguistik verwendet werden soll.[4]

[4]Vgl. bspw. die komplexen Fortsetzer von uridg. *dg^{ih}em- 'Erde': Toch. A. *tkaṃ* 'Erde', gr. χθών 'Erde', slav. *zemljā́* 'Erde',

Konsekutive *indels*

Der traditionelle dynamische Programmieralgorithmus setzt *indels* immer dort ein, wo diese die geringsten Kosten verursachen. Für linguistische Zwecke ist es mitunter jedoch ratsam, den Algorithmus zu zwingen, *indels* eher zu clustern, als sie getrennt voneinander einzusetzen, da beim Vergleich von Wörtern die tatsächlich ähnlichen Sequenzen mitunter eher in einzelnen Morphemen anzutreffen sind, als im ganzen Wort.

Kondrak (2002) beschreibt ausgehend von Gotoh (1982), wie der Algorithmus sensitiv für konsekutive *indels* gemacht werden kann: Zusätzlich zur grundlegenden Matrix werden zwei weitere Matrizzen angelegt, welche die besten Alignments aufnehmen, die konsekutive *indels* aufweisen. Tabelle 8 gibt ein Beispiel für die unterschiedlichen Alinierungsergebnisse der traditionellen und der erweiterten Fassung für gr. δίδωμι 'ich gebe' vs. russ. *dam* 'ich gebe'.

Traditionell	d	i	d	oː	m	i
	d	a	-	-	m	-
Gotoh-Erweiterung	d	i	d	oː	m	i
	-	-	d	a	m	-

Tabelle 8: Konsekutive *indels* in der Sequenzalinierung

Kompressionen und Expansionen

Zuweilen mag es sinnvoll sein, zwei Elemente einer Sequenz direkt, und ohne Ansetzen eines *indels* mit einem Element einer weiteren Sequenz gegenüberzustellen. Dies ist programmiertechnisch ähnlich einfach zu implementieren wie die Erweiterung des dynamischen Programmieralgorithmus um Transpositionen (vgl. Kondrak 2002, 39f, Oommen 1995), die in Abschnitt 2.1 angesprochen wurde, mit der Ausnahme, dass nicht für beide Sequenzen zwei Segmente einander gegenübergestellt werden, sondern je zwei Segmente einem einzigen. Tabelle 9 gibt ein Beispiel für die Alinierung von dt. [tʰɔxtʰɐ] 'Tochter' vs. schw. [dɔtʰɐ] 'Tochter'.[5] Da die Kompression-Expansions-Erweiterung des Algorithmus jeweils ei-

Levenshtein	d	ɔ	-	tʰ	ɐ
	tʰ	ɔ	x	tʰ	ɐ
Oommen-Erweiterung	d	ɔ	tʰ		ɐ
	tʰ	ɔ	xtʰ		ɐ

Tabelle 9: Kompression und Expansion in der Sequenzalinierung

ne explizite Alternative zum Einfügen von *indels* darstellt, sollte in diesem Zusammenhang noch kurz darauf hingewiesen werden, dass für eine sinnvolle Alinierung von Sequenzen mit Algorithmen, die Kompressions-Expansions-Optionen enthalten, der Vergleichsfunktion eine große Bedeutung zukommt. Nur wenn diese sehr genau austariert ist, lässt sich hier verhindern, dass der Computer scheinbar wahllos irgendwelche Segmente zusammenfasst und diese mit anderen Segmenten matcht.

lat. *humus* 'Erde, Erdboden', ai. *kṣáḥ*, die von einem simplen Alinierungsalgorithmus, der Transpositionen einschließt, wohl schwer erfasst werden könnten.

[5] Als Kosten für die Expansions-Kompressions-Operation wurde 1 angesetzt, wenn Identität in einem Substring bestand, ansonsten 1000, um wahlloses Zusammenfassen von Segmenten zu größeren Einheiten zu verhindern.

Lokale Alinierung

Anstatt zwei Sequenzen komplett zu alinieren, kann man den Algorithmus auch dahingehend verändern, eine Subsequenz minimaler Distanz zu ermitteln. Kondrak (2002:35-37) beschreibt die Erweiterung des dynamischen Programmieralgorithmus für das Ermitteln lokaler Alinierungen, jedoch basiert seine Vergleichsfunktion auf Ähnlichkeiten, im Gegensatz zu den gängigen distanzbasierten Vergleichsfunktionen, da Ähnlichkeiten nach Meinung von Kondrak besser geeignet seien, sinnvolle lokale Alinierungen zu produzieren. Ich bin von Kondraks Argumenten bezüglich der Vorteile von Ähnlichkeiten anstelle von Distanzen nicht vollständig überzeugt, insbesondere auch deshalb, weil die Ähnlichkeiten, welche Kondrak definiert, nicht auf Distanzen abbildbar sind, und somit erst durch umständliche Formeln (vgl. Downey *et al.* 2008) wieder in eine Ähnlichkeit zurückverwandelt werden müssen, wenn man die Ergebnisse von Kondraks Algorithmus für phylogenetische Studien verwenden möchte. Tabelle 10 gibt ein von Kondrak übernommenes Beispiel für lokale Alinierung von Cree *āpakosīs* 'Maus' vs. Fox *wāpikonōha* 'Maus'.

Levenshtein	-	ā	p	a	k	o	s	ī	s	-
	w	ā	p	i	k	o	n	ō	h	a
Lokale Alinierung		ā	p	a	k	o	sīs			
	w	ā	p	i	k	o	nōha			

Tabelle 10: Globale vs. lokale Alinierung

Die Frage, inwieweit lokale Alinierung tatsächlich sinnvoll ist, steht in engem Zusammenhang mit der Frage, welche Daten man überhaupt vom Computer alinieren lassen möchte. Wahrscheinlich wird man bessere Ergebnisse erzielen, wenn man auf den Schritt der lokalen Alinierung verzichtet und vorbereinigte Daten verwendet. [6]

2.2 Erweiterungen der Vergleichsfunktion

Abgesehen von den algorithmischen Erweiterungen ist es für linguistische Zwecke, sei es in der Dialektologie oder der historischen Linguistik, natürlich naheliegend, an der Vergleichsfunktion selbst anzusetzen, um bessere Ergebnisse bei der Alinierung zu erziehlen. Dies ist insbesondere in der historischen Linguistik unvermeidbar, wie sich anhand von kognaten Wortpaaren wie engl. [dɔ:tʰɚ˞] 'Tochter' gegenüber gr. [θigatɛra] 'Tochter' demonstrieren lässt: Geht man von der IPA-Schreibweise aus, so teilen beide Wörter nicht ein einziges Segment, was dazu führt, dass derartige Wortpaare in der Computeranalyse, wenn man als Vergleichsfunktion die nur zwischen Identität und Unterschiedlichkeit diskriminierende Levenshtein-Distanz zugrundelegt, mit maximalen Distanzwerten und unsinnigen Alignments belegt werden (vgl. Tabelle 11).

Kleinere Modifikationen der Vergleichsfunktion, wie beispielsweise die geringere Bestrafung der Gegenüberstellung von Konsonanten mit Konsonanten oder Vokalen mit Vokalen im Gegensatz zur hohen Bestrafung der Gegenüberstellung von Vokalen mit Konsonanten, können die Ergebnisse des Algorithmus in bestimmten Fällen bereits erheblich verbessern, was im Beispiel aus Tabelle 11 jedoch eher Einfluss auf

[6] In vielen Fällen läuft man Gefahr, den Algorithmen zu viel zuzumuten, und nicht zu bedenken, dass Menschen, die Alinierungen vornehmen, viel mehr Vorwissen über die jeweiligen Sprachen (Morphologie, Semantik, Lautkorrespondenzen, usw.) mitbringen. Will man wie das Testset von Covington (1996) vorsieht, einen Algorithmus an Wortpaaren wie lat. *mōns* 'Berg' vs. engl. *mountain* 'Berg' testen, so ist es unsinnig, die lateinische Nominativform alinieren zu lassen und dem Computer die entscheidende Information über den zugrunde liegenden Stammauslaut vorzuenthalten.

θ	i	g	a	t	ɛ	r	a
d	ɔ:	t^h	ɤˠ	-	-	-	-

Tabelle 11: Das Versagen simpler Vergleichsfunktionen bei der Alinierung phonetischer Sequenzen

das für die beiden Sequenzen ermittelte Distanzmaß, denn auf ihr Alignment hat, da ja beide Sequenzen mit 'CVCV' beginnen.

Beim Anpassen der Vergleichsfunktion sind dem Erfindungsgeist oder der Präzisierungswut des jeweiligen Forschers kaum Grenzen gesetzt: Theoretisch kann für jede Entsprechung phonetischer Segmente eine individuelle Distanz festgelegt werden, phonetische Segmente können zu Gruppen geclustert und anderen Gruppen gegenübergestellt werden, oder Einzelne Laute können wiederum weiter segmentiert und als solche in die Analyse einbezogen werden. Auch aus der historischen Linguistik bekannte häufig auftretende Lautwandelphänomene könnten in die Vergleichsfunktion einbezogen werden und beispielsweise viel geringer bestraft werden als weniger häufig auftretende Arten von Lautwandel.

Beim 'Auftunen' der Vergleichsfunktion sollte man jedoch immer beachten, bis zu welchem Maß ein solches Vorgehen noch praktikabel ist, und ob es überhaupt signifikant helfen kann, die Ergebnisse zu verbessern. Die Erfahrung zeigt, dass man mit der simplen Levenshtein-Distanz bereits sehr akzeptable Ergebnisse erzielen kann, selbst wenn man diese nur auf orthographische Daten anwendet. Als Beispiel zeigt Abbildung 1 eine auf der Berechnung der normalisierten Levenshtein-Distanz [7] basierte Neighbor-Joining-Analyse der germanischen Daten in Tower of Babel (vgl. Starostin 2008): Die orthographischen Signale sind trotz Unterschieden in Groß- und Kleinschreibung bspw. im Deutschen, die ja in einem simplen Stringvergleich als unähnlich gewertet werden, stark genug, um die klassische Grobgruppierung der germanischen Sprachen erkennen zu lassen. Die Tatsache, dass diese Analyse der germanischen Sprachen, die vom Computer in weniger als 10 Sekunden erstellt wird, näher an die traditionellen Auffassungen über die Entstehung der germanischen Sprachen herankommt, als die kognatenbasierte bayesianische Analyse von Gray & Atkinson (2003), obwohl alle skandinavischen und französischen Lehnwörter in diesem Falle in die Berechnung miteinflossen, zeigt, dass - wenn man keine großen Ansprüche an die Genauigkeit und den Realismus derartiger Klassifikationen stellt - bereits mit geringstem Aufwand sehr schnell bestimmte Ergebnisse erzielt werden können (vgl. hierzu auch die ähnlich simple Analyse der Datenbank von Dyen *et al.* 1997 von Serva & Petroni 2008). Beim Anspruch an derartige Klassifikationen beginnt aber das tatsächliche Problem computergesteuerter Analysen: Soll man sich mit groben Clusterungen, die in etwa der Trennung von Säugetieren und Vögeln in der Biologie entsprechen, zufrieden geben, oder hegt man die Hoffnung, mehr mit Computeranalysen erreichen zu können und nicht nur das bestätigt zu bekommen, was man ohnehin schon weiß?

2.3 Veränderung der Darstellung der Segmente

Eine Alternative zu komplexen Vergleichsfunktionen stellt die vor dem Datenabgleich vorgenommene Veränderung der Darstellung der Segmente dar. Streng genommen stellt auch die vorgezogene Änderung der Daten eine Veränderung der Vergleichsfunktion dar und ließe sich auch problemlos in dieser Funktion selbst implementieren. Programmiertechnisch gesehen ist es jedoch zumeist einfacher, eine Reihe von

[7] Die Levenshtein-Distanz zwischen zwei Strings geteilt durch die Länge des längeren der beiden Strings, vgl. Holman *et al.* (2008b)

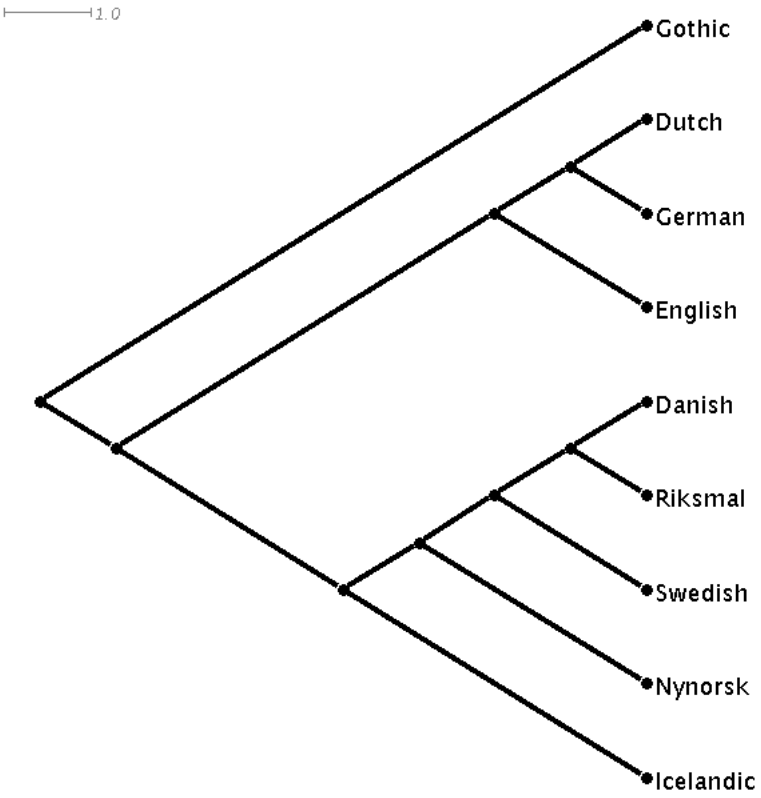

Abbildung 1: Neighbor-Joining-Analyse germanischer Sprachen auf Basis von Levenshteindistanzen

regulären Ausdrücken zu definieren, welche die Ausgangsdaten in ein bestimmtes Format verwandeln, anhand dessen dann eine mehr oder weniger traditionelle Levenshteinanalyse vorgenommen wird, als die Datentransformation in der Vergleichsfunktion anzusiedeln.

No.	**Typ**	**Beschreibung**	**Beispiel**
1	P	labiale Obstruenten	p,b,f
2	T	dentale Obstruenten	d,t,θ,ð
3	S	alveolare, postalveolare und retroflexe Frikative	s,z,ʃ,ʒ
4	K	velare und postvelare Obstruenten und Affrikaten	k,g,ʦ,ʧ
5	M	labialer Nasal	m
6	N	übrige Nasale	n,ɲ,ŋ
7	R	Trills, Taps, Flaps und laterale Approximanten	r,l
8	W	stimmhafter labialer Frikativ/Approximant und initiale gerundete Vokale	v,u
9	J	palataler Approximant	j
10	ø	Laryngale und initialer velarer Nasal	h,ɦ,ŋ

Tabelle 12: Dolgopolskys Klassifizierung von Lautwandeltypen

Als Beispiel für derartige Vorveränderungen der Segmente vor dem eigentlichen Alinieren sei meine Implementierung eines Vorschlags von Dolgopolsky (1986) genannt, welcher in seinem Artikel eine auf empirischen Studien, die leider nicht weiter explizit erläutert werden, beruhende probabilistische Klassifizierung von Lautkorrespondenzen vornimmt. Dolgopolsky unterscheidet 10 Typen von Lauten, zwischen denen Lautkorrespondenzen jeweils sehr wahrscheinlich seien (vgl. Tabelle 12). Meine Implementierung des Ansatzes für die Sequenzalinierung in der historischen Linguistik besteht im Prinzip lediglich in der Ersetzung der Ausgangszeichen (Ausgangspunkt ist eine Wiedergabe der Zeichen in IPA) in diese Lautklassen (wobei ich die von Dolgopolsky nicht klassifizierten Vokale einer eigenen Gruppe zuordne), welche dann von einem einfachen Alinierungsalgorithmus miteinander verglichen werden [8]. Auf diese Weise wird bspw. gr. [θigatɛra] als 'TVKVTVRV' wiedergegeben, engl. [dɔːtʰɚ] als 'TVTVR' und dt. [tʰɔxtʰɐ] als 'TVKTVR'. Die Ergebnisse der Alinierung sind zuweilen recht erfreulich (vgl. Tabelle 13), da bestimmte Lautwandelphänomene (Palatalisierung, Spirantisierung) explizit von Dolgopolsky addressiert wurden. Ein vollständiger Vergleich des Ansatzes mit anderen Ansätzen steht jedoch noch aus.

Interne Alinierung	T	V	K	V	T	V	R	V
	T	V	-	-	T	V	-	-
Ausgabe	θ	i	ɣ	a	t	ɛ	r	a
	d	ɔː	-	-	tʰ	ɚ	-	-

Tabelle 13: Interne und externe Darstellung der Alinierung nach Dolgopolsky

Was für die Modifizierung der Vergleichsfunktion galt, gilt auch für die Vorveränderung der Segmente: Der Phantasie des Forschers sind keine Grenzen gesetzt. Die Daten können nach phonetischen, historischen oder auch sprachfamilienspezifischen Werten umgewandelt und an die jeweiligen Vorstellungen angepasst werden.

[8] Die Vergleichsfunktion bestraft das matchen der Vokal-Klasse mit den anderen Klassen sehr hoch, während sie für das Matchen der anderen Klassen untereinander dieselben Kosten wie für die *indels* ansetzt

3 Vorstellung neuerer Ansätze von Sequenzanalysen in der historischen Linguistik

Im Folgenden werden drei neuere Ansätze in der historischen Linguistik vorgestellt, die auf Sequenzanalysen basieren: das *Automated Similarity Judgment Project* (ASJP), der Alinierungsalgorithmus von Covington (1996) und Kondraks ALINE-Algorithmus (vgl. Kondrak 2002). Es gibt noch weitere und zuweilen sogar sehr ausgefeilte Ansätze (vgl. insbesondere die RuGL^{04}-Software von Kleiweg 2009), die hier aus Platzgründen nicht angesprochen werden können. Die drei Beispiele wurden jedoch insbesondere deshalb ausgewählt, weil sie ausdrücklich in der historischen Linguistik angesiedelt werden können, während viele weitere innovative Ansätze eher dialektologischen Fragen gewidmet sind.

3.1 ASJP

Ursprünglich nahm das ASJP-Projekt die alte, bereits in Swadesh (1954) formulierte und von vielen weiteren Autoren (vgl. bspw. Ringe 1992, Baxter & Manaster Ramer 2000 und Kessler 2001) ausgebaute Idee auf, kognate Wörter probabilistisch zu ermitteln, indem basierend auf phonetischer Ähnlichkeit Basiswörter verschiedener Sprachen automatisch verglichen wurden)vgl. Brown *et al.* 2008). Für dieses Vorhaben wurde eigens eine spezielle Swadesh-Liste von 40 Basiskonzepten erstellt (vgl. die Beispiele in Tabelle 14, vgl. auch Holman *et al.* 2008b), die sich in den Voruntersuchungen als besonders stabil erwiesen hatten. Da die Untersuchung von vornherein global angelegt war, wurde ein spezielles Alphabet designt, durch welches IPA-Zeichen in vereinfachter Form wiedergegeben werden können. Inzwischen umfasst die Datenbasis des Projektes Daten über mehr als 3000 verschiedene Sprachen. Der ursprüngliche Algorithmus für den Vergleich der Sprachen wurde inzwischen jedoch aufgegeben. Stattdessen verwendet ASJP die einfache Levenshtein-Distanz, die allerdings in verschiedenerlei Hinsicht normalisiert wird (vgl. Holman *et al.* 2008a). Die Sequenzanalysen des ASJP-Projektes basieren also auf der Vormodifizierung der Segmente, während der ursprüngliche Algorithmus ohne Veränderungen übernommen wird.

Sprache	Bedeutung	**IPA**	**ASJP-Code**
engl.	'das'	ðis	8is
engl.	'Mund'	mauθ	mau8
engl.	'Zunge'	tʰəŋ	th~3N
dt.	'Fisch'	fiʃ	fiS
engl.	'Zahn'	tuːθ	tu8
dt.	'Schwester'	ʃwɛstʰɐ	Swasth~a

Tabelle 14: Das universale Alphabet des ASJP-Projektes

3.2 Covington

Der Algorithmus von Covington (1996) dient - im Gegensatz zu dem vorwiegend an Distanzen interessierten Ansatz des ASJP-Projektes - explizit der Alinierung von Sequenzen in der historischen Linguistik. Covington selbst weist den dynamischen Programmieralgorithmus als inadäquat für die Programmierung zurück und bevorzugt stattdessen eine umfangreiche Baumsuche, welche aus allen Alinierungsmöglichkeiten zweier Sequenzen die jeweils besten ermittelt. Ich stimme mit Kondrak (2002:32f) überein, dass diese Art von Implementierung weit über das Ziel hinausgeht und ab einer bestimmten Sequenzlänge nicht

mehr berechenbar ist. Darüber hinaus lässt sich Covingtons Algorithmus jedoch Problemlos im Rahmen des dynamischen Programmieralgorithmus implementieren. Er erfordert die Erweiterung des traditionellen Algorithmus um konsekutive *indels* (vgl. Abschnitt 2.1), sowie eine relativ komplexe Vergleichsfunktion. Covingtons Vorgaben für die Alinierung sind in Tabelle 15 wiedergegeben (vgl. Covington 1996, 487).

Penalty	**Conditions**
0	Exact match of consonants or glides (w, y)
5	Exact match of vowels (reflecting the fact that the aligner should prefer to match consonants rather than vowels if it must choose between the two)
10	Match of two vowels that differ only in length, or i and y, or u and w
30	Match of two dissimilar vowels
60	Match of two dissimilar consonants
100	Match of two segments with no discernible similarity
40	Skip preceded by another skip in the same word (reflecting the fact that affixes tend to be contiguous)
50	Skip not preceded by another skip in the same word

Tabelle 15: Covingtons 'Evaluationsmetrik' für den Alinierungsalgorithmus

3.3 Kondraks ALINE

Der Algorithmus von Kondrak (2002) ist der bei weitem komplizierteste von den hier besprochenen. Als Erweiterungen zum allgemeinen dynamischen Programmieralgorithmus beinhaltet er Kompressionen und Extensionen (vgl. Abschnitt 2.1), lokale Alinierung (vgl. Abschnitt 2.1) und eine komplexe, auf dem Vergleich phonetischer Merkmale beruhende Vergleichsfunktion. Im Gegensatz zu den traditionellen Implementierungen des dynamischen Programmieralgorithmus, die allesamt eine Distanz zwischen zwei Sequenzen ermitteln, berechnet Kondraks Algorithmus eine Ähnlichkeit, welche nicht direkt in eine Distanz überführt werden kann. Implementiert ist der Algorithmus in C++ und in dieser Form auch frei verfügbar, gleichzeitig gibt es eine Probeversion im Internet. Der Sourcecode wird ausführlich in Kondrak (2002) beschrieben, wenn es auch möglich ist, dass in der Zwischenzeit bestimmte Änderungen vorgenommen wurden. Die Daten müssen in einem ASCII-Alphabet eingegeben werden, das jedoch voll IPA-kompatibel ist. Die Berechnung der Werte für die Alinierungsmatrix ist relativ kompliziert: Gleichen Vokalen wird die Ähnlichkeit 15 zugestanden, gleichen Konsonanten die Ähnlichkeit 35. Unterscheiden sich Konsonanten oder Vokale in bestimmten Merkmalen, so werden diese aufgrund einer Liste von Merkmalssalienzen von dem Identitätswert abgezogen: [k] und [k] erhalten demnach die Ähnlichkeit 35, [k] und [k^h] die Ähnlichkeit 35-5=30, [k] und [g] ergeben 35-10=25 und [k^h] und [g] 35-10-5=20. Tabelle 16 listen die Merkmalssalienzen, welche ursprünglich als Standardwerte für ALINE zugrunde gelegt wurden (vgl. Kondrak 2002, 55).

Kondraks Algorithmus ist programmiertechnisch sehr weit ausgereift und basiert auf einer ausgeklügelten Ähnlichkeitsfunktion. Problematisch bei seinem Ansatz scheint mir allerdings die Aufgabe der Distanz zugunsten der Ähnlichkeit zu sein, weil dadurch das 'metrische Terrain' verlassen wird, welches einerseits für phylogenetische Studien eine große Rolle spielt und andererseits als Maß der Unterschiedlichkeit viel leichter zu interpretieren als ein nicht direkt in Distanzen überführbares Ähnlichkeitsmaß.

Syllabic	5	Place	40
Voice	10	Nasel	10
Lateral	10	Aspirated	5
High	5	Back	5
Manner	50	Retroflex	10
Long	1	Round	5

Tabelle 16: Merkmalssalienzen in ALINE

Ferner stellt sich die Frage, ob es ausreicht, bzw. ob es überhaupt realistisch ist, phonetische Ähnlichkeit als Grundlage für Sequenzanalysen in der historischen Sprachwissenschaft zu verwenden.

4 Arbeitsstand und Ausblick

Seit etwa 3 Monaten arbeite ich an einer Pythonimplementierung verschiedener Algorithmen zur automatischen Sequenz- und Mengenanalyse mit dem Arbeitstitel 'EvoClass'. Die Idee ist es, ein Programm zu entwickeln, dass speziell auf die Bedürfnisse historischer Linguisten abgestimmt ist und eine Vielzahl von Funktionen, sowie verschiedene Algorithmen verschiedenster Autoren zur Verfügung stellt. Geplant sind, neben der Implementierung verschiedener Sequenzanalyse (Covington 1996, Kondrak 2002, ASJP, usw.), auch die automatische Durchführung von Formatierungsaufgaben zu implementieren, wie sie bereits von der STARLING-Software (vgl. Starostin 1993) zu Teilen bereitgestellt werden, also die automatische Zerlegung von lexikostatistischen Daten in Kognatensets ('etymologische Wörterbücher' im rudimentärsten Sinne), sowie verschiedene Validierungsmethoden einzubeziehen, die von großer Bedeutung für das Testen der Datenkonsistenz sind (Bootstrap, Randomisierung von Daten, usw.). Das Ziel ist es, ein Programm zu entwickeln, das nicht nur einzelne Formen der Datenanalyse ermöglicht (wie die ASJP-Software, Kondraks ALINE, Kleiwegs RuGL^{04}, STARLING), sondern eine Vielzahl von Algorithmen und Methoden bereitstellt, deren Ergebnisse schnell und effizient verglichen werden können. Abbildung 2 gibt einen Überblick über die derzeit geplante Struktur des Programmes.

Eine Reihe von Algorithmen sind derzeit bereits implementiert. Zu diesen zählen:

- der Alinierungsalgorithmus von Covington (1996)
- die einfache und die normalisierte Levenshtein-Distanz (vgl. Holman *et al.* 2008b
- die Damerau-Levenshtein-Distanz
- die Einbeziehung von Kompressionen und Expansionen
- die Einbeziehung konsekutiver *indels*
- die Ausgabe von Alinierungen zwischen Sequenzen für alle Arten von Distanzen
- eine erweiterte Levenshtein-Distanz, welche sensitiv ist für Konsonant-Vokal-Distinktionen

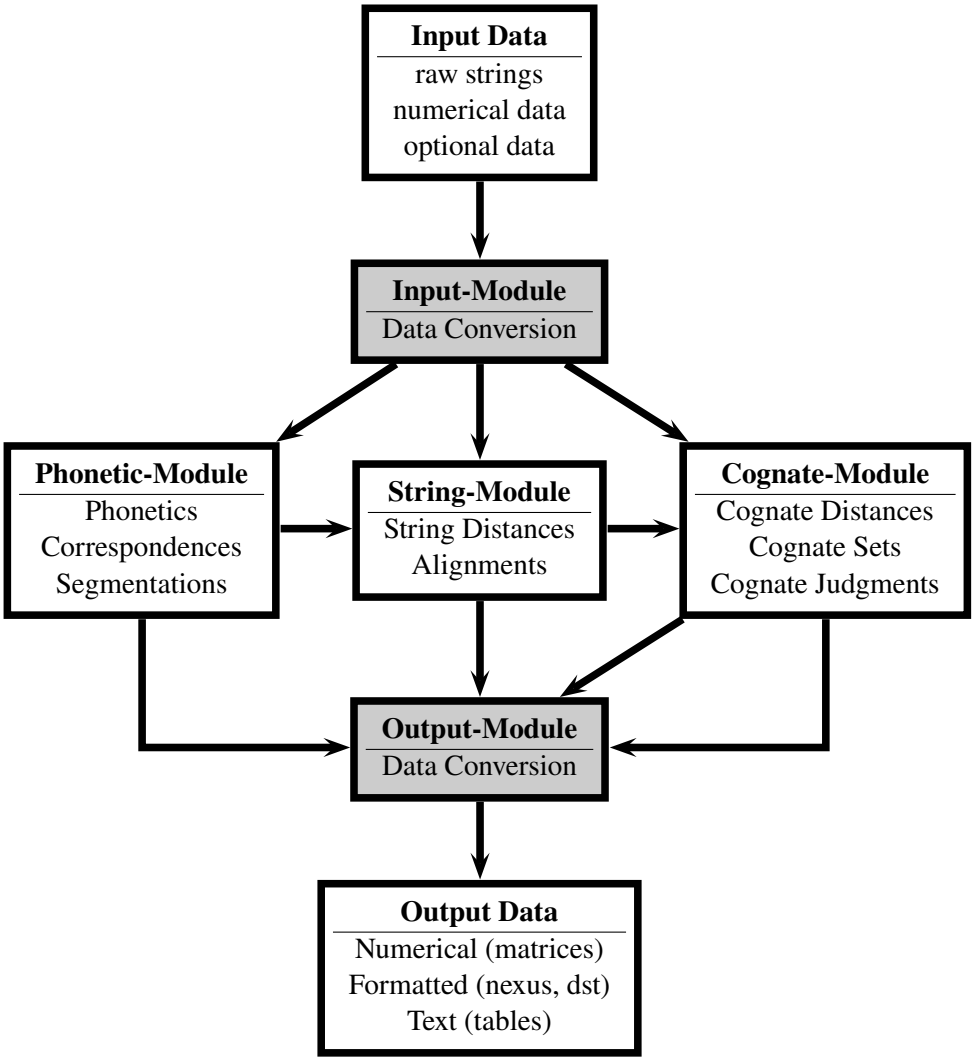

Abbildung 2: Struktur des Programms

- ein Alinierungsalgorithmus, der auf der Vorbereinigung phonetischer Daten in Anlehnung an Dolgopolsky (1986) basiert [9]
- eine Distanzanalyse, welche die Vorbereinigung der Daten gemäß dem ASJP-Projekt zugrunde legt

Geplant sind für die nahe Zukunft die Implementierung des Ansatzes von Kondrak (2002), die Entwicklung alternativer individueller Distanzen für phonetische Einheiten, mit denen sich Sven Sommerfeld

[9] Dieser Ansatz wurde nach meinem Wissen bisher nur von Baxter & Manaster Ramer (2000) explizit verwendet, allerdings nur zur probabilistischen Ermittlung von Lautkorrespondenzen. Erste Alinierungsergebnisse sind - wenn man von der allgemeinen Problematik derartiger automatischer Analysen absieht - recht vielversprechend, müssen jedoch umfangreicher getestet werden.

derzeit beschäftigt, sowie die Implementierung *n*-gramm-basierter Analysen. Implementiert sind desweiteren automatische Formatierungen, wie sie auch mit STARLING vorgenommen werden können (Kognatenanalysen, Ermittlung traditioneller lexikostatistischer Distanzen), eine Computersimulation des Sprachwandelansatzes von Holm (2000) und erste Ansätze für die probabilistische Ermittlung von Lautkorrespondenzen (vgl. hierzu insbesondere Ringe 1992, Baxter & Manaster Ramer 2000 und Kessler 2001), welche gleichzeitig als Parameter für die Vergleichsfunktion des dynamischen Programmieralgorithmus verwendet werden können (erste Tests wurden diesbezüglich schon durchgeführt, jedoch mussten als Ausgangspunkt die alles in allem nicht sehr überzeugenden Daten von Dyen *et al.* 1997 genommen werden, da keine alternativen Daten zur Verfügung standen).

Um die entsprechenden Methoden hinreichend testen zu können, ist es ferner von großer Bedeutung, ein umfangreiches Testset zu erstellen. Das von Covington (1996) verwendete und von Kondrak (2002) übernommene Testset weist eine Vielzahl von Fehlern (meist falsche Kognaten, oder eine falsche,bzw. nicht aussagekräftige Auswahl der Grundformen) auf, die behoben werden müssen, um die verschiedenen Algorithmen verlässlich vergleichen und ihre Stärken und Schwächen korrekt beurteilen zu können.

References

Baxter, William H., & Alexis Manaster Ramer. 2000. Beyond lumping and splitting: Probabilistic issues in historical linguistics. In *Time depth in historical linguistics*, ed. by Colin Renfrew, April McMahon, & Larry Trask, Papers in the prehistory of languages, 167–188. Cambridge: The McDonald Institute for Archaeological Research.

Bilu, Yonatan, Pankaj K. Agarwal, & Rachel Kolodny. 2006. Faster algorithms for optimal multiple sequence alignment based on pairwise comparisons. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3.408–422.

Brown, Cecil H., Eric W. Holman, Søren Wichmann, & Vivekacysouw Velupillai. 2008. Automated classification of the world's languages: a description of the method and preliminary results. *STUF, Berlin* 61.285–308.

Covington, Michael A. 1996. An algorithm to align words for historical comparison. *Association for Computational Linguistics* 22.481–496.

Damerau, Fred J. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7.171–176.

Dolgopolsky, A. B. 1986. A probabilistic hypothesis concerning the oldest relationships among the language families of northern eurasia. In *Typology Relationship and Time*, ed. by T. L. Shevoroshkin, Vitaly V.; Markey, Notes on Linguistics, 27–50. Karoma Publisher, Inc. Originally published in Russian as "Gipoteza drevnejščego rodstva jazykov Severnoj Evrazii (problemy fonetičeskich sootvetstvij)" in 1964.

Downey, Sean S., Brian Hallmark, Murray P. Cox, Peter Norquest, & Stephen Lansing. 2008. Computational feature-sensitive reconstruction of language relationships: Developing the aline distance for comparative historical linguistic reconstruction. *Journal of Quantitative Linguistics* 15.340–369.

Dyen, Isidore, Joseph B. Kruskal, & Paul Black, 1997. Comparative Indo-European database: File IEdata1.

Gotoh, Osamu. 1982. An improved algorithm for matching biological sequences. *Journal of Molecular Biology* 162.705 – 708.

Gray, Russell D., & Quentin D. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature* 426.435–439.

Heeringa, Wilbert J., Peter Kleiweg, Charlotte Gooskens, & John Nerbonne. 2006. Evaluation of string distance algorithms for dialectology. In *Linguistic Distances Workshop at the joint conference of International Committee on Computational Linguistics and the Association for Computational Linguistics*, ed. by John Nerbonne & E. Hinrichs, 51–62, Sydney.

Holm, Hans J. 2000. Genealogy of the main indo-european branches applying the separation base method. *Journal of Quantitative Linguistics* 7.73–95.

Holm, Hans J. 2007. The new arboretum of indo-european "trees": Can new algorithms reveal the phylogeny and even prehistory of indo-european? *Journal of Quantitative Linguistics* 14.167–214.

Holman, Eric W., Søren Wichmann, Cecil H. Brown, Viveka Velupillai, André Müller, & Dik Bakker. 2008a. Advances in automated language classification. In *Quantitative Investigations in Theoretical Linguistics*, ed. by Antti Arppe, Kaius Sinnemäki, & Urpu Nikann, 40–43. Helsinki: University of Helsinki.

Holman, Eric W., Søren Wichmann, Cecil H. Brown, Viveka Velupillai, Andr´e Müller, & Dik Bakker. 2008b. Explorations in automated lexicostatistics. *Folia Linguistica* 20.116–121.

Kessler, Brett. 2001. *The significance of word lists: Statistical tests for investigating historical connections between languages*. Dissertations in linguistics. Stanford, Calif: CSLI Publications.

Kleiweg, Peter, 2009. RuG/L^{04}. Software for dialectometrics and cartography. Distributed by the Author. Rijksuniversiteit Groningen. Faculteit der Letteren.

Kondrak, Grzegorz, 2002. *Algorithms for language reconstruction: Dissertation*. Toronto: University of Toronto dissertation.

Kondrak, Grzegorz. 2005. N-gram similarity and distance. In *Proceedings of the Twelfth International Conference on String Processing and Information Retrieval (SPIRE 2005)*, 115–126, Buenos Aires.

Kruskal, Joseph B. 1983. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review* 25.201–237.

Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10.707–710.

Makaev, E. A. 1977. *Obščaja teorija sravnitel'nogo jazykoznanija [Common theory of comparative linguistics]*. Moskva: Nauka.

Morgenstern, Burkhard, Andreas Dress, & David Werner, Thomas. 1996. Multiple dna and protein sequence alignment based on segment-to-segment comparison. *Proceedings of the National Acadamy of Science, USA* 93.12098–12103.

Needleman, Saul B., & Christan D. Wunsch. 1970. A gene method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48.443–453.

Oommen, B. John. 1995. String alignment with substitution, insertion, deletion, squashing, and expansion operations. *Inf. Sci. Inf. Comput. Sci.* 83.89–107.

Ringe, Donald A., Jr. 1992. On calculating the factor of chance in language comparison. *Transactions of the American Philosophical Society* 82.1–110.

Serva, Maurizio, & Filippo Petroni. 2008. Indo-european languages tree by levenshtein distance. *EPL* 81.

Shapiro, R. 2007. Glottochonology and etymostatistics for the study of Beijing and Sichuan dialects of Mandarin Chinese. In *Aspects of comparative linguistics.*, ed. by I. S. Smirnov, volume 2 of *Orientalia et Classica. Papers of the Institute of Oriental and Classical Studies. Issue XI.*, 393–408. Moskva: Russian State University for the Humanities.

Starostin, George, 2008. Tower of babel. an etymological database project. http://starling.rinet.ru/main.html.

Starostin, Sergej Anatol'evic. 2000. Comparative-historical linguistics and lexicostatistics. In *Time depth in historical linguistics*, ed. by Colin Renfrew, April McMahon, & Larry Trask, Papers in the prehistory of languages, 223–265. Cambridge: The McDonald Institute for Archaeological Research. Translation: Peiros, Ilia. Originally published in 1989.

Starostin, Sergej Anatol'evič. 1993. Rabočaja sreda dlja lingvista (working environment for a linguist). In *Bazy dannyh po istorii Evrazii v srednie veka*, p. 7–23.

Swadesh, Morris. 1950. Salish internal relationships. *International Journal of American Linguistics* 16.157–167.

Swadesh, Morris. 1952. Lexico-statistic dating of prehistoric ethnic contacts: With special reference to North American Indians and Eskimos. *Proceedings of the American Philosophical Society* 96.452–463.

Swadesh, Morris. 1954. On the penutian vocabulary survey. *International Journal of American Linguistics* 20.123–133.

Swadesh, Morris. 1955. Towards greater accuracy in lexicostatistic dating. *International Journal of American Linguistics* 21.121–137.

Wagner, Robert A., & Michael J. Fischer. 1974. The string-to-string correction problem. *J. ACM* 21.168–173.

Wagner, Robert A., & Roy Lowrance. 1975. An extension of the string-to-string correction problem. *J. ACM* 22.177–183.