Computergestützter Sprachvergleich

Kai-Uwe Bux   Ralf Gehrke

Es ist nicht gerade eine neue Erkenntnis, daß lautliche Entwicklungen innerhalb einer Sprache in der Regel nicht auf einzelne Wörter beschränkt sind, sondern den Wortschatz in seiner ganzen Breite erfassen. Wirken in verschiedenen Teilen einer Sprachgemeinschaft unterschiedliche Lautgesetze, so entwickeln sich unterschiedliche Dialekte und schließlich entstehen verschiedene Sprachen, die aber Abkömmlinge einer gemeinsamen Grundsprache sind, aus der sie sich gemäß jeweils eigenen Gesetzmäßigkeiten entwickelt haben. Insofern sind die entstehenden Sprachen untereinander verwandte Geschwister.

Es ist eine der klassischen Aufgaben der Sprachvergleichung, unter der Annahme einer Verwandtschaft zwischen verschiedenen Sprachen durch Vergleichung der einzelsprachlichen Befunde ein möglichst genaues Bild der ihnen gemeinsamen Grundsprache zu entwerfen. Ist die Rekonstruktion erfolgreich, so kann ihr Gelingen ihrerseits als Beleg für die im voraus postulierte Hypothese genommen werden, die betrachteten Sprachen seien untereinander verwandt – so wie das fortgesetzte Scheitern von Rekonstruktionsversuchen impliziert, daß die Verwandtschaftshypothese aufzugeben ist.

Computer können dem sprachvergleichenden Wissenschaftler auf verschiedene Weise helfen. Zum einen ist es mit ihrer Hilfe möglich, an einzelnen Beispielen gefundene lautgesetzliche Hypothesen auf den ganzen Wortschatz einer Sprache, sofern dieser in maschinenlesbarer Form vorliegt, anzuwenden und mithin ihre allgemeine Gültigkeit zu testen.

Eine zweite Möglichkeit liegt darin, Computer bei der Sprachvergleichung für die Generierung lautgesetzlicher Hypothesen heranzuziehen. Im utopischen Grenzfall würde der Computer, gespeist mit Beschreibungen verschiedener Sprachen, ermitteln, welche von ihnen untereinander verwandt sind, die jeweiligen Grundsprachen rekonstruieren und die Lautgesetze angeben, welche von den Grundsprachen zu den Einzelsprachen führen, dabei die relative Chronologie der lautlichen Veränderungen finden und so-mit schließlich einen detaillierten Stammbaum inklusive aller sprachlichen Zwischenstufen für jedes einzelne Wort ausgeben – von der grundsprachlichen Wurzel bis zu den belegten Wortformen an den Blättern. Davon wird man allerdings noch eine ganze Weile bloß träumen können. Dennoch ist es diese Art des Rechnereinsatzes, die wir hier behandeln wollen.

1 Automatisierter Lautvergleich von zwei Sprachen

Das Projekt, das wir auf Anregung von Herrn Prof. Dr. Jost Gippert verfolgen, behandelt folgende Vereinfachung des oben dargestellten utopischen Traums: Gegeben zwei Sprachen – in unserem Fall Altavestisch (ältestes Altiranisch) und Vedisch (ältestes Altindisch) –, die als verwandt angenommen werden – in unserem Fall eine gut gesicherte Hypothese –, rekonstruiere lautliche Entsprechungen aus einer Liste $(\mathfrak{a}_1, \mathfrak{v}_1), (\mathfrak{a}_2, \mathfrak{v}_2), \ldots$ von Wortpaaren, die je einen avestischen und einen vedischen Partner enthalten, die beide nach sprachwissenschaftlicher Einschätzung auf ein und dasselbe grundsprachliche Wort lautgesetzlich zurückgehen.

aϑā	átha
bagā	bhágā
baodaṇtō	bódhantaḥ
baraitī	bhárati
barətū	bharatu
barātā	bhrā́tā
bauuaiṇtī	bhávanti
bauuaṭ	bhávat
baxṣ̌aitī	bhakṣati
buuaṇticā	bhúvantica

Tabelle 1: Eingabe (Liste von Entsprechungspaaren)

Tabelle 1 zeigt einen Ausschnitt der von uns herangezogenen Liste. Sie wurde von Prof. Dr. J. Gippert kompiliert und ist die einzige Stelle, an der sprachwissenschaftliche Expertise in die Analyse eingeht. Grundlage war das vollständige altavestische Korpus. Von den ca. 3000 hierin bezeugten Wortformen (Tokens) haben ungefähr die Hälfte eine Entsprechung im Vedischen, die aber nicht immer perfekt ist (z. B. durch unterschiedliche Präverbierung etc.). Es bleiben ungefähr 1000 klare Fälle. Diese Paarungen bilden die von uns verwendete Liste.

Was der Algorithmus ausgeben soll, ist eine Liste $\mathfrak{g}_1, \mathfrak{g}_2, \ldots$ von anzusetzenden Wortformen in einer hypothetischen gemeinsamen Vorläufersprache G und zwei Sätze A und V von Lautgesetzen, so daß für jede Wortform $\mathfrak{g}_i$ der konstruierten Liste die Anwendung der Lautgesetze aus A auf das Wort $A(\mathfrak{g}_i) = \mathfrak{a}_i$ aus der Inputliste führt, während sich der vedische Partner $\mathfrak{v}_i$ als $V(\mathfrak{g}_i)$ aus $\mathfrak{g}_i$ dadurch ergibt, daß die Regeln aus V angewendet werden.

Diese Aufgabenbeschreibung ist noch zu vage, als daß sie zur Grundlage der Programmierung werden könnte. Wir müssen noch angeben, welche logische Form die Lautgesetze annehmen dürfen. In einem ersten Schritt lassen wir nur allereinfachste Lautgesetze zu, die keine Kontextbedingungen beinhalten. Die Wortformen der Grundsprache werden so angesetzt, als ob ein bestimmter Laut der Grundsprache im Avestischen stets zu dieser, im Vedischen stets zu jener Lautfolge geworden sei. Dies führt natürlich zu einer relativ großen Zahl von anzusetzenden grundsprachlichen Lauten. Ziel des Algorithmus ist auf dieser Stufe, den anzusetzenden Lautbestand der Grundsprache durch geschickt gewählte Ansätze für die einzelnen Wortformen möglichst klein zu halten.

Hier gibt es zu jedem grundsprachlichen Laut g in A und V je genau ein Gesetz, das angibt, welche Lautfolgen $A(g)$ und $V(g)$ dem Laut g im Avestischen und Vedischen jeweils entsprechen. Gilt für zwei Laute g und h der Grundsprache $A(g) = A(h)$ und $V(g) = V(h)$, so lassen sich g und h in beiden Einzelsprachen nicht mehr unterscheiden. Diese Redundanz läßt sich beseitigen, indem in jeder angesetzten grundsprachlichen Wortform konsequent h durch g ersetzt wird, was den angenommenen Lautbestand der Grundsprache verkleinert. Daher kann der grundsprachliche Laut g mit dem geordneten Paar $(A(g), V(g))$ identifiziert werden. Das Ziel der ersten Stufe läßt sich mithin wie folgt formulieren:

Aufgabe 1 *Zerlege die Wortpaare aus der Inputliste so in einander entsprechende Segmente, daß die Gesamtzahl der dabei auftretenden veschiedenen*

a	ϑ	ā				
á	th	a				
b	a	g	ā			
bh	á	g	ā			
b	ao	d	a	ṇ	t	ō
b	ó	dh	a	n	t	aḥ
b	a	r	a	i	tī	
bh	á	r	a	t	i	
b	a	r	ə	t	ū	
bh	a	r	a	t	u	

ba	r	ā	t	ā				
bh	r	ā́	t	ā				
b	a	uua	i	ṇ	t	ī		
bh	á	v	a	n	t	i		
b	a	uu	a	ṭ				
bh	á	v	a	t				
b	a	x	ṣ̌	a	it	ī		
bh	a	k	ṣ	a	t	i		
b	u	u	a	ṇ	t	i	c	ā
bh	ú	v	a	n	t	i	c	a

Tabelle 2: Ausgabe in Schritt 1 (Zerlegungen der Wortpaare in einander entsprechende Segmente)

Segmentpaare (grundsprachliche Laute) minimal wird.

Tabelle 2 zeigt einen Ausschnitt der Zerlegungen, die unser erfolgreichster Algorithmus für die von uns verwendete Liste ausgibt.

Selbstverständlich gibt es auch kontextabhängige Lautentwicklungen. Diese versuchen wir in einem zweiten Analyseschritt zu erfassen, dessen Ziel es ist, den anzusetzenden Lautbestand der Grundsprache weiter zu reduzieren, indem die grundsprachlichen Laute (Phone), die in der ersten Stufe gefunden wurden, zu Gruppen (Phonemen) zusammengefaßt werden. Dabei muß für jedes Phonem stellungsbedingt vorhersagbar sein, durch welches Phon es an einer bestimmten Stelle in einem bestimmten Wort realisiert wird. Betrachten wir also die Hypothese, der grundsprachliche Lautbestand zerfalle in die Phoneme $\bar{\mathsf{g}} = \{\mathsf{g}_1, \ldots, \mathsf{g}_r\}$, $\bar{\mathsf{g}}' = \{\mathsf{g}'_1, \ldots, \mathsf{g}'_{r'}\}, \ldots$, so entspricht jedem Wort $\mathfrak{g} = \mathsf{g}_1 \mathsf{g}_2 \cdots \mathsf{g}_s$ eine phonematische Darstellung $\bar{\mathfrak{g}} = \bar{\mathsf{g}}_1 \bar{\mathsf{g}}_2 \cdots \bar{\mathsf{g}}_s$ definiert durch die Bedingung $\mathsf{g}_i \in \bar{\mathsf{g}}_i$. Eine Hypothese über die Zusammenfassung von Phonen zu Phonemen ist zu verwerfen, wenn zwei verschiedene Worte der Grundsprache in phonematischer Schreibung zusammenfallen, mit anderen Worten, wenn $\bar{\mathfrak{g}} = \bar{\mathfrak{g}}'$ für zwei verschiedene Worte $\mathfrak{g} \neq \mathfrak{g}'$ vorkommen sollte. Denn in diesem Fall können die einzelsprachlichen Formen nicht mehr vorhergesagt werden. Es wird in der zweiten Phase der Analyse also um folgendes gehen:

Aufgabe 2 *Zerlege die Menge der hypothetischen grundsprachlichen Laute (Phone) aus dem ersten Analyseschritt in möglichst wenige Teilmengen (Phoneme), so daß für je zwei Worte* $\mathfrak{g}, \mathfrak{g}'$ *der Grundsprache gilt*

$$\mathfrak{g} \neq \mathfrak{g}' \Longrightarrow \bar{\mathfrak{g}} \neq \bar{\mathfrak{g}}'.$$

Für diese Aufgabe haben wir noch keine überzeugende algorithmische Lösung gefunden, so daß wir hier keinen Beispieloutput zur Illustration angeben können.

Zusammenfassend kann man sagen, daß unserem Ansatz folgende linguistische Fiktion zugrunde liegt: Wir tun so, als ob sich der Lautwandel von einer Grundsprache zu zwei Einzelsprachen in zwei Schritten vollzieht. In einem ersten bilden sich in der Grundsprache stellungsbedingte Varianten von Phonemen, während sich in einer zweiten Stufe die verschiedenen Phone je nach Zielsprache unterschiedlich weiterentwickeln, wobei wiederum Phone zusammenfallen können. Die automatische Rekonstruktion entdeckt die beiden Stufen von den Einzelsprachen ausgehend, so daß die spätere Entwicklung im ersten, die frühere im zweiten Schritt gefunden wird.

2 Heuristik zur Lösung von Aufgabe 1

In der Darstellung von Aufgabe 1 haben wir ein Segmentpaar mit einem Laut einer hypothetischen Grundsprache identifiziert, der sich im Avestischen zur einen Komponente des Paares und im Vedischen zur anderen Hälfte entwickelt hat. Es stellt sich zunächst die Frage, ob wir dabei Entwicklungen zu 0, also den Wegfall eines grundsprachlichen Lautes im Avestischen oder Vedischen zulassen wollen oder nicht. In der Formulierung von Aufgabe 1 heißt das, ob in den Zerlegungen von Wörtern in Segmente auch leere Segmente auftreten dürfen.

Betrachtet man Wegfall von Lauten als zulässige lautliche Entwicklung, so hat Aufgabe 1 leider eine triviale Lösung. Nehmen wir an, das

Avestische habe einen Lautbestand von 45 Lauten und das Vedische habe 52 Laute. Dann können wir eine Grundsprache mit 97 Lauten annehmen, in der auf dem Weg zum Vedischen 45 Laute weggefallen sind und 52 überlebt haben, während im Avestischen gerade diese 52 weggefallen, die andern hingegen erhalten sind. Es ist außerdem ein leichtes, die grundsprachlichen Wortformen anzusetzen, die zwar deutlich länger sind als die avestischen und vedischen Formen – aber was will man erwarten, wenn in jeder der Einzelsprachen ungefähr die Hälfte aller Laute wegfallen.

Diese triviale Lösung von Aufgabe 1 ist natürlich sprachwissenschaftlich unsinnig: So kann man zwei beliebige Sprachen als verwandt hinstellen, und ein Verwandtschaftsbegriff, der immer zutrifft, führt nicht zu einer interessanten Theorie. Läßt man jedoch eine Maschine, die ja nicht über ihr Tun nachdenkt, nach einer Lösung für Aufgabe 1 suchen, so übt die triviale Lösung eine starke Anziehungskraft aus. Wir haben verschiedene Algorithmen getestet, die mit Hilfe anderer Bewertungsheuristiken von der trivialen Lösung fern bleiben sollten, doch es kam niemals ein sinnvolles Ergebnis heraus. Dieser empirische Befund ist nicht verwunderlich: Läßt man Wegfall von Lauten zu, erlaubt man leere Segmente in den Segmentpaaren, so ist Aufgabe 1 keine linguistisch sinnvolle Problemstellung, weil es eine sprachwissenschaftlich unsinnige Lösung dieser Aufgabe gibt.

Wir sehen uns mithin zu der radikalen Lösung genötigt, keine leeren Segmente zuzulassen. Das hat gravierende Konsequenzen, denn Wegfall von Lauten kommt natürlich in realen Lautentwicklungen vor, und auch epenthetische Einfügungen in bestimmten Kontexten erzeugen leere Segmente, nämlich wenn in einer der Sprachen die Einfügung nicht erfolgt ist. Darum analysiert unser Algorithmus zum Beispiel das Paar (barātā bhrā́tā) als:

ba	r	ā	t	ā
bh	r	ā́	t	ā

und nicht als:

b	a	r	ā	t	ā
bh		r	ā́	t	ā

Verbieten wir dem Algorithmus diese Analyse, so entstehen Kosten – und zwar in der Form, daß wir für die Grundsprache noch mehr Laute ansetzen müssen als durch die Beschränkung auf kontextfreie Lautgesetze ohnehin erzwungen ist. Unser bester Algorithmus führt im Fall der avestisch-vedischen Liste auf einen grundsprachlichen Lautbestand von ca. 350 Lauten. Von Hand läßt sich das Ergebnis allerdings noch weiter verbessern.

Nachdem wir Aufgabe 1 nun präziser dahingehend verstehen, daß Segmentpaare keine leeren Segmente enthalten dürfen, wissen wir im Prinzip auch, wie man diese Aufgabe lösen kann: Eine *Zerlegung* eines Wortpaares sei eine Zerlegung beider enthaltenen Wörter in gleichviele nicht-leere Segmente. Dabei entspricht das erste Segmente des avestischen Wortes dem ersten Segment des vedischen Partners etc. Eine *Konfiguration* ist die Wahl einer Zerlegung für jedes Wortpaar unserer Liste, also in unserem Fall eine Reihe von ungefähr 1000 Zerlegungen. Nun probiere man einfach alle Konfigurationen durch und wähle eine solche, bei der möglichst wenig verschiedene Paare entsprechender Segmente vorkommen.

Eine leichte Überschlagsrechnung zeigt, daß es viel zu viele Konfigurationen gibt, als daß man sie alle durchrechnen könnte. Ein durchschnittliches Wortpaar gestattet vielleicht um die 25 verschiedene Zerlegungen. Bei 1000 Wortpaaren sind das 25^{1000} Konfigurationen – das ist eine Zahl mit fast 1400 Stellen und damit jenseits von Gut und Böse.

Die Suche nach einer guten Konfiguration in dieser Menge erfordert daher eine geschicktere Strategie, und da man in einer strukturlosen Menge nicht mehr tun kann, als sie erschöpfend zu durchsuchen, braucht eine bessere Suchstrategie eine Struktur auf dieser Menge. Wir nennen zwei Konfigurationen *benachbart*, wenn sie für alle Wortpaare bis auf eines gleiche Zerlegungen vorsehen. Durch diese Nachbarschaftsrelation wird die Menge aller Konfigurationen zu einem Raum. Jede Konfiguration ist ein Punkt darin. Die zu minimierende Anzahl verschiedener Segmentpaare fassen wir als Höhe des Punktes auf. Wir sollten uns also einen Planeten wie die Erde vorstellen, und die Aufgabe ist, den tiefsten Punkt der Oberfläche zu finden.

In DUECK & SCHEUER (1990) und DUECK et al. (1993) werden zwei gute Heuristiken zur Lösung dieses Problems vorgestellt.

- Beim *Sintflutalgorithmus* zur Minimierung denken wir uns die Erde vollständig von Wasser überflutet. Im Meer schwimmt ein Fisch. Geht das Wasser langsam zurück, bis es endlich vollständig verschwindet, so wird der Fisch schließlich irgendwo auf dem Trockenen liegen bleiben, und, so erstaunlich das auch scheinen mag, der Fisch bleibt in der Regel an einem sehr tiefen Punkt der Erdoberfläche liegen.

- Der zweite Algorithmus benutzt eine *Toleranzschwelle* T für begrenzten Aufstieg. Hier denken wir uns einen Wanderer auf der jetzt als völlig trocken vorgestellten Erdoberfläche ausgesetzt, der zwar beliebig schnell absteigen kann, aber bei jedem Schritt (von einer Konfiguration zu einem Nachbarn) höchstens T an Höhe gewinnen kann. Im Laufe der Zeit ermüdet der Wanderer und T geht gegen 0, so daß der Wanderer schließlich nicht mehr aufwärts, sondern nur noch abwärts gehen kann. So wird der Wanderer in ein sehr tiefes Tal geraten: wie Versuche zeigen, in ein deutlich tieferes, als wenn er von Anfang an nur abstiege.

Wir haben bessere Erfahrungen mit dem zweiten Verfahren erzielt, womit wir die Berichte von Dueck, Scheuer und Wallmeier bestätigen können.

Das eigentliche Problem liegt in der Höhenfunktion. Zählen wir nur die Anzahl der verschiedenen Segmentpaare, so erhalten wir offenbar stets eine ganzzahlige Höhe. Nach unserer Erfahrung ist diese Höhenfunktion jedoch nicht gut genug für die beschriebenen Algorithmen: In der Nachbarschaft eines Punktes kann man leider nicht sehen, in welcher Richtung es wirklich und auf Dauer abwärts geht.

Aus diesem Grund gewinnen wir aus der Konfiguration eine verfeinerte Höhe – wir erfinden gewissermaßen Stellen nach dem Komma. Die besten Resultate liefert eine Verfeinerung, in der auf ziemlich unanschauliche, scheinbar willkürliche Weise auch die Häufigkeiten der Segmentpaare und die Längen der involvierten Segmente eingehen.

3 Nutzen und Grenzen

Solange wir keine algorithmische Lösung von Aufgabe 2 anbieten können, ist der sprachwissenschaftliche Nutzen des Programmes sehr beschränkt. Und selbst wenn Aufgabe 2 gelöst wäre, ließe sich immer noch nicht die relative Chronologie lautgesetzlicher Entwicklungen automatisch ermitteln, da wir nur zwei Sprachen vergleichen und in der Modellierung Elision und Epenthese ausschließen.

Es wäre aber übertrieben, wenn man dem Programm jegliche wissenschaftliche Bedeutung absprechen wollte. Die Größe des anzusetzenden grundsprachlichen Alphabets, das Aufgabe 1 löst, hängt zwar stark von der Länge der Wortpaarliste ab, doch mißt sie auch den Verwandtschaftsgrad der beteiligten Sprachen – je weniger grundsprachliche Laute nötig sind, desto verwandter sind die beteiligten Sprachen.

Außerdem hat sich das Programm bereits als nützlich erwiesen, wenn auch eher unter praktischen Gesichtspunkten. Denn wo der Algorithmus gezwungen ist, Segmententsprechungen anzunehmen, die nur in einem einzigen Wortpaar vorkommen, stellt sich oft heraus, daß ein Eingabefehler in der Ausgangsliste vorliegt.

References

DUECK, Gunter and SCHEUER, Tobias (1990): Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. Journal of Computational Physics 90, p. 161 – 175.

DUECK, Gunter et al. (1993): G. D., Tobias SCHEUER, Hans-Martin WALLMEIER, Toleranzschwelle und Sintflut: neue Ideen zur Optimierung. Spektrum der Wissenschaft (1993/März), p. 42 – 51.