Modellierung von Lautwandel (Übung zur Vertiefung)

1 Einleitendes vorweg

Anmerkungen zum Lesetext

Der Text von Karttunen (1993), welcher für diese Sitzung als Lesetext vorgesehen ist, ist ein absoluter Overkill, sowohl in Bezug auf unseren bisherigen Wissensstand, als auch in praktischer Hinsicht. Das habe ich selbst erst bemerkt, als ich die Planung des Seminars bereits aufgesetzt hatte. Ich habe den Text nicht zurückgenommen, weil ich davon ausgehe, dass jeder, der sich die Mühe gemacht hat, den Text zu lesen, davon profitiert hat, sei es, dass er oder sie neue Erkenntnisse gewinnen konnte, oder, dass er oder sie sich das Gefühl erarbeitet hat, mit einem Text richtig gerungen zu haben (ohne dieses Gefühl einmal erlebt zu haben, sollte keiner die Uni abschließen). Und was die möglichen neuen Erkenntnisse betrifft, so ist es meiner Ansicht nach tatsächlich eine faszinierende Erkenntnis, die in dem Text beschrieben wird, nämlich, dass die Komplexität phonologischer Regeln (und womöglich auch von historischen Lautwandelregeln) eben nicht so groß ist, wie man normalerweise geneigt ist, anzunehmen. Denjenigen, die sich mehr für diese Fragestellungen interessieren, also für die Fragen zur Kompelexität formaler Sprachen, empfehle ich die Einführung von Durbin u. a. (1998 [2002]), welche sich zwar an Biologen richtet, aber viel einfacher zu verstehen ist, als die ursprünglichen Arbeiten von Chomsky (1959).

Darstellung von Lautwandelprozessen

In der letzten Sitzung wurde angesprochen, dass man Regeln, ähnlich denen in der Phonologie, verwendet, um Lautwandelprozesse abzubilden. Wir wollen nun in der Praxis ausprobieren, was das konkret heißen kann, indem wir versuchen, die Regeln zu ermitteln, die das Althochdeutsche zum Deutschen gemacht haben. Dafür wurden 517 Wörter automatisch ausgewählt, die in vereinfachter Orthographie im Althochdeutschen und im Deutschen vorliegen, und von denen wir davon ausgehen können, dass sie in direkter Verwandtschaftsbeziehung zueinander stehen. Die Daten wurden dabei aus Kluge (KLUGE) entnommen und nachträglich automatisch normalisiert. Unter `http://lingulist.de/lautwandel/SoundChanger.html` wurde eine Webseite (mit Modifikationen übernommen von Mark Rosenfelders SCA², abrufbar unter `http://www.zompist.com/sca2.html`) erstellt, welche Lautwandelregeln implementiert, und die Daten zur Verfügung stellt. Die Regeln für die Modellierung von Lautwandel sind dabei relativ einfach. Das Grundlegende Prinzip ist

```
R: I > O / C
```

wobei `I` für den Input-Laut steht, `0` für den Output-Laut, und `C` für den Kontext. Um die Regeln mächtiger zu gestalten, gibt es noch ein paar spezielle Kniffe, die im Folgenden gesondert beschrieben werden.

Kategorisierung von Lauten

Ein elementares Prinzip in der Darstellung von Lautwandelprozessen ist die Gruppierung von Lauten zu sinnvollen Klassen, also deren „Kategorisierung". „Sinnvoll" ist dabei ein dehnbarer Begriff, und es ist auch nach wie vor nicht klar, ob Sinnvollheit universell oder einzelsprachabhängig ist.[1] Wenn wir aber von einer Sprache ausgehen, die zwischen langen und kurzen Vokalen unterscheidet, dann könnte man beispielsweise eine Gruppierung der Vokale in lange und kurze Vokale als „sinnvoll" ansehen. Hierzu definieren wir beim Anwenden von Lautwandelgesetzen zunächst eine *Kategorie*, in diesem Fall eine für die langen, und eine für die kurzen Vokale:[2]

K1: `V=aeiou`

K2: `L=āēīōū`

Der Vorteil der Verwendung von Kategorien ist, dass man mit ihrer Hilfe Regeln viel, viel einfacher definieren kann. Wenn wir jetzt zum Beispiel alle Vokale im Auslaut eines Wortes kürzen wollen (ein durchaus möglicher Lautwandelprozess), dann müssen wir nicht mehr für jeden Vokal einzeln eine derartige Regel schreiben:

R1: `ā > a /_#`

R2: `ē > e /_#`

R3: `ī > i /_#`

R4: `ō > o /_#`

R5: `ū > u /_#`

Es genügt, wenn wir einfach die Kategorien-Label stattdessen verwenden:

R1: `V > L /_#`

Wichtig ist hierbei, dass die einander entsprechenden Elemente, also [a] und [ā], an der gleichen Stelle definiert werden: die Reihenfolge der Definition muss stets die gleiche sein, da sonst unterschiedliche Elemente miteinander ersetzt werden.

Gruppierung von Lauten

Mitunter kann es sinnvoll sein, Laute nicht fest zu kategorisieren, sondern sie lediglich für bestimmte Lautwandelprozesse zu Gruppen zusammenzufassen. Dies ist entweder dann sinnvoll, wenn der Wandel, um den es geht, idiosynkratischer Natur ist, wie bspw. die *ruki*-Regeln im vedischen Sanskrit, derzufolge [s] nach den Lauten [r, u, k] und [i] zu [ʂ] retrofligiert wird (Meier-Brügger

[1] „Sinnvoll" ist kein feststehender Terminus! In der historischen Linguistik spricht man oft von „Natürlichkeit", wobei auch das, meiner Meinung nach, kein richtig sinnvoller Terminus ist...

[2] Ich benutze hier aus technischen Gründen bewusst NICHT die IPA-Notation, obwohl dies natürlich viel konsequenter wäre. Im Moment ist der Quellcode aber noch nicht so weit, dass er vollständig mit IPA umgehen kann.

2002: 233), oder wenn man bestimmte Ersetzungsmuster kurzfristig testen will. Die Gruppierung wird hier mit Hilfe eckiger Klammern dargestellt. Die *ruki*-Regel sähe demnach wie folgt aus:

```
R s>ş/[ruki]_
```

Zu beachten ist, dass im Gegensatz zu den Kategorien die Gruppierung *nur* im Input oder im Kontext verwendet werden kann, da sie „on the fly" gebildet wird, und das Programm folglich nicht die Möglichkeit hat, die Anzahl an Gruppenmitgliedern zu zählen und somit die Zugehörigkeit zu ermitteln.

Kontext

Generell ist der Kontext recht einfach zu handhaben, indem man einfach den Tiefstrich _ verwendet, um auf den ursprünglichen Laut zu verweisen, der geändert wird, und dann rechts und links davon unterschiedliche, für den Kontext relevante Elemente einsetzt, wobei das Symbol # für den Wortanfang bzw. das Wortende steht, und das Symbol Ø, bzw. alternativ auch der einfache Bindestrich - als Platzhalter für eine Insertion oder Deletion gilt. Will man also beispielsweise den Laut [i] am Ende eines jeden Wortes tilgen, so schreibt man einfach:

```
R i > - /_#
```

Und wenn man am Anfang eines jeden Wortes, dass mit [st] beginnt, ein [e] einfügen, dann schreibt man:

```
• - > e /#_st
```

Auch hier gilt es immer, die vielfältigen Möglichkeiten der Kategorisierung zu beachten! Wenn wir zum Beispiel alle Wörter, die mit [s] und nachfolgendem Konsonanten beginnen, durch eine Epenthese von [e] verschönern wollen, dann ist es am besten, vorher eine spezielle Kategorie für Konsonanten festzulegen:

```
K: C=bcdfghjklmnprstvw
```

Danach können wir dann bequem unsere Regel definieren:

```
R: - > e /#_sC
```

Wichtig ist auch die Möglichkeit, *fakultative* Kontexte zu bestimmen, indem man die Elemente einklammert. Zum Beispiel fügt die folgende Regel nur dann ein [e] am Wortanfang ein, wenn unmittelbar auf das [s] ein Vokal folgt, oder ein Konsonant, gefolgt von einem Vokal:

```
R: - > e / #_s(C)V
```

Auch ein Phänomen, wie der Umlaut kann mit Hilfe der fakultativen Klammern dargestellt werden. Wenn wir vom einfachen Fall ausgehen, dass nur das [u] zum [ü] wird, wenn es von einem [i] in der nächsten Silbe gefolgt wird, können wir bspw. schreiben:

```
R: u > ü / _(C)Ci
```

Die Klammern können übrigens nur in den Kontextausdrücken verwendet werden.

Preparse

Manchmal klappt es einfach nicht richtig mit dem Lautwandel, egal wie sehr man sich bemüht. Zuweilen liegt das auch daran, dass manche Regeln eben doch ein wenig komplexer sein können. Bisweilen sind es aber auch einfach idiosynkratische Einzelheiten, die man am besten ignoriert. Für diesen Fall bietet das Online-Programm noch einen weiteren Regelsatz an, mit dessen Hilfe einfache Ersetzungen *ohne Kontext* („Preparse" genannt) angewendet werden können. Es empfielt sich, hier vor allem diejenigen Wörter umzuwandeln, welche notorisch irregulär sind. Das Programm wendet dabei zunächst diese sehr einfachen und sehr konkreten Regeln an und fängt erst daraufhin an, den richtigen Lautwandel zu „simulieren". Um diese Regeln bequemer zu gestalten, darf dabei das Wortgrenzenzeichen # in der Inputregel eingesetzt werden. Will man alle Fälle des Präfixes *bi-* im Althochdeutschen beispielsweise durch seine neuhochdeutsche Entsprechung *be-* ersetzen, schreibt man einfach:

P: `#bi > be`

Tips und Tricks

Ersetzungen sind nicht nur an einfache Laute gebunden! Man kann auch ein ganzes Wort durch ein anderes Wort ersetzen. Dabei folgt man einfach den Regeln, die wir schon aufgesetzt haben (nur dass man dann keinen Kontext braucht):

R: `mäh > muh`

Regeln müssen also nicht immer und nicht zwangsläufig so abstrakt sein, wie zu Beginn besprochen. Vor allem in Fällen von irregulären Lautwandeltypen ist es daher sinnvoll, eine Regel in einer solch idiosynkratischen Form zu notieren, als durch eine Anwendung einer abstrakten Regel einen Wandel zu bewirken, der gar nicht stattfand.

Die **Reihenfolge** ist entscheidend! Wenn man mit einer ersten Regeln alle [s] in ein [z] verwandelt, kann man in einer weiteren Regel nicht alle [s] entsprechend der *ruki*-Regel retrofligieren, da ja kein [s] mehr vorhanden ist! Man muss ich also gute Gedanken machen, in welcher Reihenfolge die Regeln auftreten sollen.

Kommentieren und **Strukturieren** ist wichtig und sinnvoll, wenn man gute Regeln erstellen will. Kommentieren heißt hier: explizit aufschreiben, worauf sich eine Regel bezieht, und zwar im Text, wobei Kommentare mit Hilfe des Prozentzeichens als solche gekennzeichnet und vom Programm ignoriert werden, genauso wie leere Zeilen, die keine Ausdrücke enthalten. Kommentieren ist eine sehr, sehr wichtige Sache beim Schreiben von Programmcode. Auch beim praktischen Erstellen von Lautwandelregeln kann es sehr nützlich sein.

2 Aufgabe

Die Aufgabe ist einfach zu erklären, aber vielleicht weniger einfach zu erledigen: Versuche, mit Hilfe von möglichst *wenigen* Lautwandelregeln ein mög-

lichst gutes Ergebnis (mehr als 50% der Daten sollten erklärt werden können!) zu finden. Dies ist das sogenannte Prinzip von „Okhams Rasiermesser", welches besagt, dass der kürzeste Lösungsweg alternativen Lösungswegen vorzuziehen ist (auch wenn es nicht immer heißen muss, dass der kürzeste Weg auch der richtige Weg ist!). Dieses Prinzip ist in historischen Linguistik nahezu ein Standard, und wir wollen versuchen, diesem Ideal so nahe wie möglich zu kommen. Wichtig für ein gutes Funktionieren ist dabei eine wohlüberlegte Auswahl der Kategorien, und natürlich eine saubere Auswahl der Regeln. Auf der Webseite sind bereits Beispiele vorgegeben, die aber nur der Illustration dienen. Im Großen und Ganzen sollte das Programm problemlos funktionieren. Wenn nicht, bitte sofort per Email bei mir melden.

Die Ergebnisse, also die Kategorien und die Regeln, zusammen mit der Auswertung, bitte ich per Email an mich zu senden (`mattis.list@uni-marburg.de`), wobei der Betreff „Hausaufgabe 1" heißen sollte (und nicht anders!). Die Regeln, Kategorien, und Ergebnisse sollten ferner entweder als einfache Text-Datei übermittelt werden, oder als LibreOffice oder Word-Datei. PDF bietet sich in diesem Fall nicht an, da man aus PDF-Dateien nicht so leicht kopieren und einfügen kann. Am einfachsten ist es hierbei, von der im Programm enthaltenen „export"-Funktion Gebrauch zu machen. Wenn man auf den entsprechenden Button drückt, nachdem die Ergebnisse bearbeitet wurden, liefert dieser alle relevanten Zusammenfassungen, einschließlich Regeln, Kategorien, Rewrites, und der Evaluierung. Die Aufgaben sollten mir spätestens am Dienstag vor der nächsten Seminarsitzung zugesandt werden. Ansonsten werde ich sie leider nicht bearbeiten können.

Literatur

Chomsky, N. (1959). "On certain formal properties of grammars". *Information and Control* 2, 137–167.

Durbin, R., S. R. Eddy, A. Krogh und G. Mitchinson (1998 [2002]). *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. 7. Aufl. Cambridge: Cambridge University Press.

Karttunen, L. (1993). "Finite-state constraints". In: *The last phonological rule: Reflections on constraints and derivations*. Hrsg. von J. Goldsmith. London: University of Chicago Press, 173–194.

Kluge, F., Begr. (2002). *Etymologisches Wörterbuch der deutschen Sprache*. Fortgef. von E. Seebold. 24. Aufl. Berlin: de Gruyter.

Meier-Brügger, M. (2002). *Indogermanische Sprachwissenschaft*. Unter Mitarb. von M. Fritz und M. Mayrhofer. 8. Aufl. Berlin und New York: de Gruyter.