Automatisierte metrische Analyse lateinischer Dichtung

Ulrich Klauer
Lunds universitet/
Georg-August-Universität Göttingen
ulrich.klauer@stud.uni-goettingen.de

Abstract

This paper describes a prototype system for the automated metrical analysis of classical Latin verse. Even though the Latin writing system not at all guarantees a perfect rendering of the features relevant for the metric in a verse, it is quite well possible to at least narrow down the results to a few candidate solutions. Use of a dictionary should greatly improve the performance and deliver a unique solution most of the time.

1 Einführung

Dichtung zeichnet sich dadurch gegenüber Prosa aus, daß in ihr der sprachliche Ausdruck durch bestimmte Regeln eingeschränkt ist. Sie können sich etwa auf die zulässige Abfolge von betonten und unbetonten Silben beziehen, wie in den Versformen der meisten modernen europäischen Sprachen, oder auf die Kombination bestimmten Wörter, wie etwa im Reim; daher bezeichnet man die Sprachform in Gedichten auch als „gebunden".

In der lateinischen Dichtung der Antike bezogen sich solche dichterischen Formen überwiegend auf die Abfolge unterschiedlich langer Silben. (Wir bezeichnen heute lange Silben als schwer, kurze als leicht; das vermeidet Verwirrung, weil schwere Silben sowohl lange als auch kurze Vokale enthalten können, leichte Silben allerdings nur kurze.) Diese Form der Versbildung heißt quantitative Metrik. Ein zulässiges Muster für einen Vers kann etwa folgendermaßen aussehen: –⏑⏑–⏑⏑–⏑⏑–⏑⏑–×. Dabei steht – für eine schwere, ⏑ für eine leichte und × für eine beliebige Silbe; ⏑⏑ steht für wahlweise eine schwere oder zwei leichte Silben. Dieses Versmaß heißt (daktylischer) Hexameter.

Wie auch in den meisten modernen Sprachen ist die lateinische Orthographie alles andere als vollkommen: Lange und kurze Vokale werden mit denselben Buchstaben wiedergegeben, sind also nicht zu unterscheiden; Diphthonge werden mit zwei Buchstaben geschrieben, die aber auch für getrennte Laute stehen können (etwa *a*-*e*); das Zeichen *i* steht sowohl für den Vokal *i* als auch den Konsonanten *j*. Dazu kommen zahlreiche weitere Variationsmöglichkeiten etwa in der Silbenbildung, die sich in der Schrift nicht widerspiegeln.

Trotzdem ist es erfahrungsgemäß für Menschen relativ einfach möglich, auch unbekannte Verse beim ersten Lesen überwiegend korrekt zu skandieren, also metrisch zu analysieren. Dies gilt selbst dann, wenn viele Vokabeln unbekannt sind und also keine Informationen aus Vorwissen ergänzt werden können. Es sollte daher möglich sein, auch einen Computer mit Heuristiken zu versehen, wie Menschen sie in dieser Situation anwenden, und ihn damit Verse automatisch analysieren zu lassen.

2 Umsetzung

Für die Realisierung des Projekts wurde Haskell als Programmiersprache gewählt; seine algebraischen Datentypen und seine bequeme Speicherverwaltung machen es einfach, auch komplexere Datenstrukturen zu realisieren.

Das Programm betrachtet jeweils nur einzelne Verse. Die Analyse erfolgt in fünf Schritten; jede Verarbeitungsstufe erhält nur die Ausgabe

des vorangegangenen Schritts. Gibt es in einem Schritt mehrere Interpretationsmöglichkeiten – etwa *ae* als Diphthong oder als zwei getrennte Vokale –, so werden sämtliche Möglichkeiten betrachtet und an die nächste Stufe weitergegeben. Weniger wahrscheinliche Möglichkeiten, in diesem Fall die Interpretation als zwei getrennte Vokale, werden dabei mit „Strafpunkten“ belegt. Dies erlaubt es, äußerst unrealistische Interpretationen später auszufiltern.

1. Zunächst wird jedes Wort einzeln in Laute zerlegt; da ein Laut bei weitem nicht immer einem Buchstaben entspricht, vereinfacht dies die weitere Verarbeitung. Zugleich wird für jeden Laut vermerkt, ob es sich um einen Vokal bzw. Diphthong oder eine von verschiedenen Konsonantengruppen handelt. In diesem Schritt wird nicht versucht, zwischen langen und kurzen Vokalen zu unterscheiden: Da ein Vokalbuchstabe gleichermaßen für einen langen wie einen kurzen Vokal stehen kann, würde sonst nur nutzlos die doppelte Zahl von Lösungskandidaten erzeugt werden. Für die Zerlegung wird gewöhnliches Parsing mit rekursivem Abstieg verwendet.

2. Anschließend werden aus den Lauten Silben gebildet. Die lateinischen Silbenbildungsregeln sind relativ einfach: Von mehreren Konsonanten, die zwischen zwei Vokalen stehen, gehören alle bis auf den letzten zur ersten Silbe; gibt es nur einen Konsonanten, gehört dieser dagegen zur zweiten Silbe. Eine Kombination aus einem Plosiv und bestimmten Dauerlauten, etwa *tr*, wird dabei meist wie ein einzelner Konsonant betrachtet (sog. „muta cum liquida“). Auch für diesen Schritt reicht ein Parser mit rekursivem Abstieg, mit etwas Nachbearbeitung, aus.

3. Nun werden die bisher isoliert betrachteten Wörter wieder zusammengefaßt. Im allgemeinen genügt es, sie einfach zu verketten; allerdings treten an den Kontaktstellen der Wörter Konsonanten zum folgenden Wort über, wenn es mit einem Vokal beginnt. Treffen dort zwei Vokale unmittelbar aufeinander, so verschwindet zudem meist eine der Silben, es können aber ausnahmsweise auch beide erhalten bleiben (*Hiat*). Dies wird beim Zusammenfügen beachtet.

4. Nachdem jetzt die Silben im Vers vollständig bekannt sind, wird in diesem Schritt versucht, ihr Gewicht zu bestimmen. Silben, die einen Diphthong enthalten oder auf einen Konsonanten enden, sind stets schwer; Silben, auf deren Vokal ohne dazwischen stehende Konsonanten der Vokal der nächsten Silbe folgt, sind meist leicht. Bei den verbleibenden Silben hängt das Gewicht ausschließlich von der Länge des Vokals ab, die ohne Wörterbuch nicht bekannt ist, und kann daher nicht bestimmt werden.

5. Abschließend wird durch *pattern matching* gegen die in Frage kommenden Versmaße geprüft, welche Lösungskandidaten eines der Versmaße realisieren könnten. Dabei werden viele Kandidaten verworfen; so kann etwa die Silbenfolge schwer–leicht–schwer in einem Hexameter nicht auftreten.

Das Ausfiltern unrealistischer Kandidaten ist am besten nach Schritt 3 möglich, wenn die Strafpunkte für den gesamten Vers vorliegen.

3 Ergebnisse

Das Programm wurde mit rund hundert Versen aus der „Aeneis“ von Vergil (durchgehend Hexameter) und den „Amores“ von Ovid (abwechselnd Hexameter und Pentameter) erprobt, die während der Entwicklung des Programms nicht getestet worden waren.

Nur bei einem Vers gelang keine metrische Analyse; die korrekte Interpretation hatte so viele Strafpunkte, daß sie als zu unwahrscheinlich verworfen wurde. In den übrigen Fällen wurde jeweils die korrekte Analyse gefunden. Allerdings konnten dabei nur in rund einem Achtel der Fälle alle anderen Lösungskandidaten ausgeschlossen werden; in den übrigen Fällen blieben mehrere Möglichkeiten offen, so daß die korrekte nicht identifiziert werden kann. In einzelnen Fällen blieben zwar bis zu vierzig falsche Kandidaten übrig, doch ist dies die Ausnahme: Immerhin bei der Hälfte aller Verse waren es nicht mehr als zwei.

Eine deutliche Verbesserung dürfte der Einsatz eines Wörterbuchs bringen, in dem Vokallängen und Diphthonge verzeichnet sind. Ob-

wohl auch dann noch nicht alle Mehrdeutigkeiten beseitigt sind – etwa bei Wörtern wie *malum*, die mit kurzem und langem *a* vorkommen, oder dem Hiat –, dürfte schon ein weniger umfangreicher Grundwortschatz die Analyse beinahe perfekt machen.

Will man weiterhin auf ein sicher aufwendig anzulegendes Wörterbuch verzichten, dürften sich die Verbesserungsmöglichkeiten dagegen auf kleinere Änderungen am Strafpunktesystem beschränken. Möglicherweise ist es aber noch möglich, bessere Heuristiken für die Deutung von *i* als *i* oder *j* zu finden: In seiner jetzigen Form hält das Programm häufig noch einen Vokal für möglich, wo ein menschlicher Leser intuitiv längst abgewinkt hätte.

4 Dank

Für das Programm habe ich die Parser-Infrastruktur adaptiert, die Lennart Andersson (Lund) in seinem Papier „Parsing with Haskell“ beschreibt. – Die Anregung, lateinische Metrik mit EDV zu behandeln, verdanke ich Marcus Deufert (jetzt Leipzig).