Institut für Theoretische Informatik
Leibniz Universität Hannover

# Wortschatzanalyse lateinischer Autoren

von Constantin-Carino Zühlke
Matrnr.: 3072850

Erstprüfer: Prof. Dr. Heribert Vollmer
Zweitprüfer: Dr. Arne Meier

09.12.2018

# Erklärung der Selbstständigkeit

Hiermit erkläre ich, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.

Diese Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

| | |
|---|---|
| Datum | Unterschrift |

# Vorwort

Das Ziel dieser Arbeit ist eine Wortschatzanalyse dreier römischer Autoren auf Basis der Werke *Tusculanae disputationes* (Gespräche in Tusculum) von Cicero, *Historiae* (Historien) von Tacitus und einer Sammlung von 116 Gedichten oder *Carmina* von Tacitus. Somit liegt je ein Text philosophischer Natur (Cicero), geschichtlicher Natur (Tacitus) und poetischer Natur (Catull) vor.

Um die Wortschatzanalyse durchzuführen ist natürlich maschinelle Hilfe von Nöten. Es ist also ein Programm notwendig, dass die gegebenen Texte einliest und die gebeugten Worte im Text ihren Grundformen zuordnen kann. Dieses Programm stellt den Kern dieser Arbeit dar. Es soll aber nur den Anfang des Weges zu einem mächtigeren Programm darstellen, das eine syntaktische oder gar semantische Analyse lateinischer Texte ausführen kann.

Sicherlich ist nicht jeder der lateinischen Sprache mächtig, weswegen die ersten Kapitel sich mit einer kleinen Übersicht befassen.

# Inhaltsverzeichnis

# Kapitel 1

# Überblick über die lateinische Sprache

Die lateinische Sprache war die Sprache des römischen Reiches und nahm durch die Ausbreitung dessen enormen Einfluss auf viele europäische Sprachen, wie wir sie heute kennen. Im Mittelalter wurde Latein aufgrund der Christianisierung und der Tatsache, dass Latein die Sprache der Kirche war, zur Sprache der Gelehrten. Obwohl zu diesem Zeitpunkt die Sprache schon als tot galt, wurden ihr immer noch neue Worte beigefügt. Heutzutage wird Latein weiterhin an Schulen gelehrt; ein Latinum gilt für ausgewählte Studiengänge (z.B. Theologie und Archäologie) als Voraussetzung. Weiterhin wird es von Liebhabern, häufig in der Musik, am Leben erhalten.

Wie jeder Sprache liegt auch der lateinischen eine Grammatik zugrunde und obwohl eine ausführliche Erläuterung den Rahmen dieser Arbeit sprengen würde, ist es doch von Vorteil, sich einen Überblick über die Regeln und Systeme, die die Flexion beschreiben, zu schaffen.

### 1.0.1 Kasus

Im Grunde genommen verhält sich Latein in vielen Aspekten ähnlich dem Deutschen. Es gibt neben den vier aus dem Deutschen bekannten Kasus zusätzlich den Ablativ und bei näherer Betrachtung den Vokativ und den Lokativ.

Der Vokativ ist eine direkte Anrede an Personen und wird daher nur mit Eigennamen verwendet. Da dieser weiterhin fast ausschließlich den Formen des Nominativ entspricht und nur bei männlichen Substantiven der zweiten Deklination eine andere Wortendung hervorruft - Worte die auf -us enden nehmen -e an, Worte die auf -ius enden nehmen -i an (*Et tu, mi fili Brute?*) - ist der Vokativ ein seltenes Phänomen.

Der Lokativ gibt den Ort eines Geschehens an und tritt daher ausschließlich bei Ortsnamen auf. Hinzu kommt, dass der Lokativ von anderen Kasus überdeckt wird, also keine eigenen Wortendungen hervorruft. Aus diesem Grund ist der Lokativ, zumindest für eine Wortschatzanalyse, hinfällig.

Der lateinische Ablativ (*ab-latus* → weg-tragen) ist kein Ablativ im eigentlichen Sinn (räumliche und zeitliche Trennung), sondern übernimmt weitere Funktionen wie z.B. Mittel/Werkzeug (Instrumentativ) oder in manchen Situationen auch Ortsangaben (Lokativ). Der Ablativ ist sehr facettenreich und kann bei der Übersetzung ins Deutsche nur mit Hilfe von Präpositionen umschrieben werden. Der Ablativ ist sehr mächtig.

Nominativ, Genitiv, Dativ und Akkusativ verhalten sich wie im Deutschen.

### 1.0.2 Numerus

Analog zum Deutschen existieren nur Singular und Plural.

### 1.0.3 Genus

Das Lateinische kennt drei Geschlechter: Das Männliche, das Weibliche und das Neutrum. Diese verhalten sich wie im Deutschen; das grammatische Geschlecht von Personen entspricht dem natürlichen. Ausnahmen hierbei bilden Lehnworte, deren Beugung so naheliegender vollzogen werden kann. Ein Beispiel hier ist der Name Adam, der nach der ersten Deklination gebeugt wird, die eigentlich nur Feminina enthält.

**Anmerkung:**

Wie schon erwähnt liegt der Fokus der Arbeit auf Worterkennung und Wortschatzanalyse; Syntax und andere grammatikalische Phänomene der lateinischen Sprache sind hier also irrelevant und werden daher auch nicht erläutert.

## 1.1 Die Deklinationen

Die lateinische Sprache kennt fünf Deklinationen. Substantive und Adjektive werden auf Basis ihrer Stammlaute in die jeweiligen Deklinationen eingeteilt, die daher auch ihre Namen beziehen:

- Die erste Deklination oder auch a-Deklination,
- Die zweite Deklination oder auch o-Deklination,

- Die dritte Deklination oder auch Konsonantische Deklination mit zusätzlich misch-stämmigen und i-stämmigen Worten,
- Die vierte Deklination oder auch u-Deklination,
- Die fünfte Deklination oder auch e-Deklination.

## 1.2 Die Konjugationen

Es existieren fünf Konjugationen. Verben werden auf Basis ihres Stammlauts in die jeweiligen Konjugationen eingeteilt, sodass auch hier die Bezeichnungen abgeleitet werden:

- Die erste Konjugation mit a-Stämmen,
- Die zweite Konjugation mit e-Stämmen,
- Die dritte Konjugation mit i-Stämmen,
- Die vierte Konjugation mit konsonantischen Stämmen,
- Die fünfte Konjugation mit konsonantischen Stämmen, die ein -i- im Präsensstamm aufweisen.

# Kapitel 2

# Flexion

In diesem Kapitel wird anhand von Beispielen die Flexion lateinischer Worte präsentiert. Bei Substantiven und Adjektiven besteht diese einzig aus Beugung der Endung eines Wortes nach Kasus, Numerus und Genus, wobei jede Deklination ihre eigenen Beugungsschemata aufweist. Da bei einigen Worten die Grundform, also die Nominativ-Singular-Form, von jenen Schemata abweichen kann, wird in einem Wörterbuch grundsätzlich immer mindestens auch die Genitivform mit angegeben. Das Beugen von Substantiven und Adjektiven bezeichnet man auch als **deklinieren**.

Verben hingegen werden wie im Deutschen nach Person, Zahl, Diathese, Tempus und Modus gebeugt. Im Gegensatz zum Deutschen können in der lateinischen Sprache mehr Formen ohne Benutzung von Hilfsverben erzeugt werden (*ivisset* - er wäre gegangen, *ibo* - ich werde gehen). Zusätzlich besitzen fast alle Verben außer einem Präsensstamm auch einen Perfektstamm und eine Partizipform, welche sich nicht pauschal aus dem Präsensstamm ableiten lassen. Diese werden in einem Wörterbuch mit angegeben. Das Beugen von Verben bezeichnet man auch als **konjugieren**.

## 2.1 Substantive

Die lateinische Sprache kennt eine Vielzahl unterschiedlichster Substantive. Obwohl mit den fünf Deklinationen eine Reihe von Bildungsschemata vorliegen, existieren viele Ausnahmen, die sich nicht ganz an diese Muster halten. In dieser schriftlichen Ausführung werden allerdings nur die archetypischen Bildungsmuster dargelegt.

Zusätzlich lassen sich deklinationsübergreifende Regeln aufstellen:

- Neutra nehmen (wie im Deutschen) im Akkusativ immer die Nominativform an.

- Die Vokativformen sind gleich den Nominativformen (mit Ausnahme der Maskulina der zweiten Deklination).

### 2.1.1 Erste und Zweite Deklination

Da die a-Deklination nur Feminina und die o-Deklination keine dieser enthält, werden hier die ersten beiden Deklinationen zusammen aufgeführt.

| Numerus | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Genus | Fem. | Mask. | Neutr. | Fem. | Mask. | Neutr. |
| Nominativ | Via | Servus | Coenum | Viae | Servi | Coena |
| Genitiv | Viae | Servi | Coeni | Viarum | Servorum | Coenorum |
| Dativ | Viae | Servo | Coeno | Viis | Servis | Coenis |
| Akkusativ | Viam | Servum | Coenum | Vias | Servos | Coenos |
| Ablativ | Via | Servo | Coeno | Viis | Servis | Coenis |
| Vokativ | Via | Serve | Coenum | Viae | Servi | Coena |

#### Anmerkung

Der Vokativ von Maskulina die im Nominativ auf -ius enden (zum Beispiel Namen wie Claudius oder Flavius) lautet -i.

### 2.1.2 Dritte Deklination

Die dritte Deklination ist die mit Abstand diffizilste. Sie umfasst Worte mit konsonantischem Stamm, i-Stamm und einer Art Mischstamm dieser. Es ist schwierig, diese drei Typen von einander abzugrenzen, da sich diese einerseits stark ähneln, andererseits die lateinische Sprache sich im Laufe der Zeit gewandelt hat und so Worte von einem Typ in den anderen übergeflossen sind. Anders als die ersten beiden Deklinationen enthält die dritte Deklination Substantive aller Genera.

Worte mit konsonantischem Stamm nehmen für alle Genera die gleichen Endungen an, mit Ausnahme der Nominativ- und Akkusativ-Plural-Endungen bei Neutra.

| Numerus | Singular | | Plural | |
|---|---|---|---|---|
| Genus | Fem. | Neutr. | Fem. | Neutr. |
| Nominativ | Merces | Corpus | Mercedes | Corpora |
| Genitiv | Mercedis | Corporis | Mercedum | Corporum |
| Dativ | Mercedi | Corpori | Mercedibus | Corporibus |
| Akkusativ | Mercedem | Corpus | Mercedes | Corpora |
| Ablativ | Mercede | Corpore | Mercedibus | Corporibus |

Zu den i-stämmigen Worten gehören Neutra und eine Handvoll Feminina, bei denen zwei Formen für den Nominativ und Akkusativ Plural auftauchen.

| Numerus | Singular | | Plural | |
|---|---|---|---|---|
| Genus | Fem. | Neutr | Fem. | Neutr. |
| Nominativ | Turris | Mare | Turres/-is | Maria |
| Genitiv | Turris | Maris | Turrium | Marium |
| Dativ | Turri | Mari | Turribus | Maribus |
| Akkusativ | Turrim | Mare | Turres/-is | Maria |
| Ablativ | Turri | Mari | Turribus | Maribus |

Die Worte der Mischklasse weisen Endungen von sowohl konsonantisch-stämmigen als auch i-stämmigen Worten auf.

| Numerus | Singular | Plural |
|---|---|---|
| Genus | Fem. | Fem. |
| Nominativ | Navis | Naves |
| Genitiv | Navis | Navium |
| Dativ | Navi | Navibus |
| Akkusativ | Navem | Naves |
| Ablativ | Nave/-i | Navibus |

### 2.1.3 Vierte Deklination

Die vierte Deklination kennt ein Schema für Maskulina & Feminina und eins für Neutra.

| Numerus | Singular | | Plural | |
|---|---|---|---|---|
| Genus | Mask. | Neutr. | Mask. | Neutr. |
| Nominativ | Casus | Genu | Casus | Genua |
| Genitiv | Casus | Genus | Casuum | Genuum |
| Dativ | Casui | Genu | Casibus | Genibus |
| Akkusativ | Casum | Genu | Casus | Genua |
| Ablativ | Casu | Genu | Casibus | Genibus |

### 2.1.4 Fünfte Deklination

Die fünfte Deklination enthält hauptsächlich feminine Substantive; die Formen bilden sich unabhängig vom Genus.

| Numerus | Singular | Plural |
|---|---|---|
| Genus | Fem. | Fem. |
| Nominativ | Res | Res |
| Genitiv | Rei | Rerum |
| Dativ | Rei | Rebus |
| Akkusativ | Rem | Res |
| Ablativ | Re | Rebus |

## 2.2 Adjektive

Die Flexion von Adjektiven erfolgt analog zu der von Substantiven. Sie werden dabei so gebeugt, dass Kasus, Numerus und Genus von Substantiv und Adjektiv übereinstimmen; man spricht von KNG-Kongruenz. Obwohl es fünf Deklinationen gibt, werden Adjektive nur anhand der ersten drei dekliniert. Adjektive tauchen in der Grundstufe Positiv und in den Steigerungsstufen Komparativ und Superlativ auf.

### 2.2.1 Positiv

Im Positiv besitzen Adjektive die Endungen aus sowohl den ersten beiden als auch aus der dritten Deklination. Auch hier treten Maskulina mit Nominativformen auf -er auf.

| Numerus | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Genus | Mask. | Fem. | Neutr. | Mask. | Fem. | Neutr. |
| Nominativ | durus | dura | durum | duri | durae | dura |
| Genitiv | duri | durae | duri | durorum | durarum | durorum |
| Dativ | duro | durae | duro | doris | doris | doris |
| Akkusativ | durum | duram | durum | duros | duras | duros |
| Ablativ | duro | dura | duro | duris | duris | duris |
| Vokativ | dure | dura | - | duri | durae | - |

| Numerus | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Genus | Mask. | Fem. | Neutr. | Mask. | Fem. | Neutr. |
| Nominativ | niger | nigra | nigrum | nigri | nigrae | nigra |
| Genitiv | nigri | nigrae | nigri | nigrorum | nigrarum | nigrorum |
| Dativ | nigro | nigrae | nigro | nigris | nigris | nigris |
| Akkusativ | nigrum | nigram | nigrum | nigros | nigras | nigros |
| Ablativ | nigro | nigra | nigro | nigris | nigris | nigris |

**Anmerkung:**

Nicht alle Adjektive auf -er verlieren das -e- (z.B. *liber, libera, liberum*).

Die Adjektive, die nach der dritten Deklination gebildet werden, lassen sich in einen der drei Typen einendig, zweiendig und dreiendig unterteilen. Die Bezeichnungen geben an, wie viele unterschiedliche Nominativ-Singular-Formen das jeweilige Adjektiv besitzt, wobei zweiendige sich eine Form für das männliche und das weibliche Geschlecht teilen. Da die Nominativformen auch hier abweichen können, wird bei einendigen Adjektiven die Genitivform mit angegeben. Außer in der Grundform gibt es zwischen den drei Typen keine Unterschiede bei der Flexion.

| Numerus | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Genus | Mask. | Fem. | Neutr. | Mask. | Fem. | Neutr. |
| Nominativ | celer | celeris | celere | celeres | celeres | celeria |
| Genitiv | celeris | celeris | celeris | celerium | celerium | celerium |
| Dativ | celeri | celeri | celeri | celerium | celerium | celerium |
| Akkusativ | celerem | celerem | celere | celeres/-is | celeres/-is | celeria |
| Ablativ | celeri | celeri | celeri | celeribus | celeribus | celeribus |

### 2.2.2 Komparativ

Der Komparativ stellt die erste Steigerungsform dar. Er wird gebildet, indem an den Stamm des Adjektivs die Endung -ior für Maskulina & Feminina und -ius für Neutra angehängt wird. Dekliniert wird nach der dritten Deklination.

| Numerus | Singular | | Plural | |
|---|---|---|---|---|
| Genus | Mask. & Fem. | Neutr. | Mask. & Fem. | Neutr. |
| Nominativ | durior | durius | duriores | duriora |
| Genitiv | durioris | durioris | duriorum | duriorum |
| Dativ | duriori | duriori | durioribus | durioribus |
| Akkustativ | duriurem | durius | duriores | duriora |
| Ablativ | duriore | duriore | durioribus | durioribus |

### 2.2.3 Superlativ

Der Superlativ stellt die höchste Steigerungsform dar. Dieser wird gebildet, indem die Endungen -issimus, -issima und -issimum an den Stamm angehängt werden (*durus - durissimus, durissima, durissimum*). Adjektive die auf -er enden, bilden den Superlativ mit -rimus, -rima und -rimum (*miser - miserrimus, miserrima, miserrimum*). Ein paar Adjektive die auf -lis enden, bilden den Superlativ mit -limus, -lima und -limum (*facilis - facillimus, facillima, facillimum*). In jedem dieser Fälle werden die Superlativformen dann wie Adjektive im Positiv anhand der ersten beiden Deklinationen gebildet.

```
Tabelle entfällt.
```

## 2.3 Verben

Verben können in der lateinischen Sprache eine Vielzahl an unterschiedlichen Formen annehmen, da sie nach Person (erste Person und zweite Person), Zahl (Singular und Plural), Diathese (Aktiv und Passiv), Tempus (Präsens, Imperfekt, Perfekt, Plusquamperfekt, Futur und Perfekt Futur) und Modus (Indikativ, Konjunktiv und Imperativ) konjugiert werden. Obwohl es fünf Konjugationsklassen gibt, werden Verben der Fünften quasi gleich denen der Dritten gebeugt, sodass sich insgesamt vier Bildungsschemata für Verben in der lateinischen Sprache ergeben.

### 2.3.1 Konjugieren des Wortes *amare*

Aufgrund der großen Anzahl von Formen, die ein einzelnes Verb annehmen kann, werden hier nur die Formen der ersten Konjugation aufgeführt. Es folgt die komplette Flexion des Wortes *amare*.

`Indikativ:`

| **Aktiv** | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | amo | amas | amat | amamus | amatis | amant |
| Imperfekt | amabam | amabas | amabat | amabamus | amabatis | amabant |
| Futur | amabo | amabis | amabit | amabimus | amabitis | amabunt |
| Perfekt | amavi | amavisti | amavit | amavimus | amavistis | amaverunt |
| Plu.Perf. | amaveram | amaveras | amaverat | amaveramus | amaveratis | amaverant |
| Perf.Fut. | amavero | amaveris | amaverit | amaverimus | amaveritis | amaverint |
| **Passiv** | Singular | | | Plural | | |
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | amor | amaris | amatur | amamur | amamini | amantur |
| Imperfekt | amabar | amabaris | amabatur | amabamur | amabamini | amabantur |
| Futur | amabor | amaberis | amabitur | amabimur | amabimini | amabuntur |

**Anmerkung:**

Die Perfekt-, Plusquamperfekt- und Perfekt-Futur-Formen werden im Passiv mit dem Partizip *amatus* und der entsprechend flektierten Form des Hilfsverbs *esse* gebildet.

`Konjunktiv:`

| **Aktiv** | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | amem | ames | amet | amemus | ametis | ament |
| Imperfekt | amarem | amares | amaret | amaremus | amaretis | amarent |
| Perfekt | amaverim | amaveris | amaverit | amaverimus | amaveritis | amaverint |
| Plu.Perf. | amavissem | amavisses | amavisset | amavissemus | amavissetis | amavissent |
| **Passiv** | Singular | | | Plural | | |
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | amer | ameris | ametur | amemur | amemini | amentur |
| Imperfekt | amarer | amareris | amaretur | amaremur | amaremini | amarentur |

**Anmerkung:**

Auch hier wird der Passiv im Perfekt und im Plusquamperfekt mit dem Partizip *amatus* und der entsprechenden Form von *esse* gebildet. Außerdem kennt die lateinische Sprache kein Futur im Konjunktiv.

`Imperativ:`

| **Aktiv** | Singular | | | Plural | | |
|---|---|---|---|---|---|---|
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | - | ama | - | - | amate | - |
| Futur | - | amato | amato | - | amatote | amanto |
| **Passiv** | Singular | | | Plural | | |
| Person | 1.Pers. | 2.Pers. | 3.Pers | 1.Pers. | 2.Pers. | 3.Pers |
| Präsens | - | amare | - | - | amamini | - |
| Futur | - | amator | amator | - | - | amantor |

`Sonstige Formen:`

| | Aktiv | | | Passiv | | |
|---|---|---|---|---|---|---|
| | Präsens | Perfekt | Futur | Präsens | Perfekt | Futur |
| Infinitiv | amare | amavisse | amaturus esse | amari | amatus esse | amatum iri |
| Partizip | amans | - | amaturus | - | amatus | amandus |

| | Nominativ | Genitiv | Dativ | Akkusativ | Ablativ |
|---|---|---|---|---|---|
| Gerundium | amare | amandi | amando | amandum | amando |
| Supinum | - | - | - | amatum | amatu |

**Anmerkung:**

Das Wort *amare* eignet sich für eine komplette Flexion über alle Formen, da es einen Präsensstamm, einen Perfektstamm und eine Partizipform besitzt. Verben die sich nicht gänzlich konjugieren lassen, werden **defektiv** genannt.

### 2.3.2 Irregulär konjugierte Verben

Wie für die lateinische Sprache üblich gibt es für jede Regel auch reichlich Ausnahmen, die hier kurz vorgestellt werden. Diese Verben heißen **irregulär**, da sie keiner Konjugationsklasse angehören und somit eigene Bildungsschemata aufweisen.

| Infinitiv | Präsens | Perfekt | Partizip |
|---|---|---|---|
| esse | sum | fui | futurus |
| posse | possum | potui | - |
| ire | eo | ivi | itum |
| velle | volo | volui | - |
| nolle | nolo | nolui | - |
| malle | malo | malui | - |
| ferre | fero | tuli | latum |
| fieri | fio | factus sum | - |
| edere | edo | edi | esum |
| dare | do | dedi | datum |

### 2.3.3 Deponentien

Deponentien sind Verben, die nur im Passiv erscheinen, allerdings aktivisch übersetzt werden. Sie sind also defektiv. Dies bereitet beim Konjugieren aber keine Probleme.

## 2.4 Pronomina

In der lateinischen Sprache finden sich eine Menge Pronomina. Von den Personalpronomen *ego* & *tu* über das Reflexivpronomen *se* zu einer Fülle an Demonstrativpronomen wie zum Beispiel *is, ea, id*, stehen viele Möglichkeiten offen, Sätze zu konstruieren. Keines dieser Pronomina wird jedoch regulär flektiert. So ergeben sich zu viele Tabellen, um sie hier aufzuführen.

## 2.5 Adverbien

Adverbien werden aus dem zugehörigen Adjektiv gebildet und nicht dekliniert. Sie lassen sich allerdings steigern.

- Adjektive, die in -us, -a, -um enden, bilden das Adverb mit -e (*durus* → *dure*),
- Adjektive, die in -is enden, bilden das Adverb mit -(i)ter (*hilaris* → *hilariter*),
- Adjektive, deren Stamm in -nt endet, bilden das Adverb mit -er (*vehemens, vehementis (gen.)* → *vehementer*).

Der Komparativ wird wie bei Adjektiven gebildet; an den Adverbstamm wird -ius angehängt (*dure* → *durius*). Da Adverbien nicht flektiert werden, gibt es auch keine extra Form für Neutra (wie *durior* beim Komparativ des Adjektivs).

Der Superlativ eines Adverbs wird gebildet, indem an den Stamm der Superlativform des Adjektivs ein -e angehängt wird (*facillimus* → *facillime*).

## 2.6 Erklärung zu der Quelle

An dieser Stelle möchte ich anmerken, dass ich dieses Kapitel nicht mit dem aus dem Lateinunterricht verbliebenen Wissen hätte schreiben können. Die hier präsentierten Informationen über die lateinische Sprache entstammen in weiten Teilen der Wikipedia, genauer dem deutschen und dem englischen Artikel zur lateinischen Grammatik (zuletzt abgerufen am 09.12.2018).

Dort finden sich neben den Flexionstabellen, die hier entfallen sind, weitere, tiefer gehende Informationen.

# Kapitel 3

# Der Programmablauf

Dieses Kapitel behandelt den Programmablauf im Groben. Um diesen in allen seinen Facetten zu begreifen, ist es notwendig, den ein oder anderen Blick in den Quellcode zu werfen. Der Programmablauf lässt sich in neun Schritte unterteilen:

1. Aufbereiten des zu analysierenden Textes
2. Parsen des Wörterbuchs
3. Pipelining der Zeileneinträge
4. Flexion
5. Nachträgliches Einfügen von Sonderformen
6. Eintragen in die Hashmap
7. Bekanntheit der Worte prüfen
8. Worte nach Bekanntheit sortieren
9. Ergebnisse in eine Textdatei schreiben

## 3.1 Aufbereiten des zu analysierenden Textes

Das Programm beginnt mit der Aufbereitung des Textes, für den eine Wortschatzanalyse durchgeführt werden soll. Diese verläuft wie folgt:

1. Sonderzeichen, Satzzeichen, Zahlen, etc. werden durch Whitespaces ersetzt.

2. Der Text wird an Whitespacecharakteren gesplittet. Es entsteht so ein String-Array, das jedes Wort einzeln hält.

3. Majuskel werden durch Minuskel ersetzt.

4. Alle 'j' werden durch 'i' ersetzt.

5. Die Worte werden Zeilenweise in eine neue Textdatei geschrieben.

## 3.2 Parsen des Wörterbuchs

Der Aufbau des Wörterbuchs ist eigentlich simpel. Ein Zeileneintrag beginnt mit '#', gefolgt von den lateinischen Grundformen. Je nach Wortart können mehrere Formen vorliegen, die dann durch ein Komma und ein Whitespace getrennt sind.

Auf die lateinischen Worte folgen zwei weitere Whitespaces und darauf die Metainformationen. Diese beginnen mit einem Wortart-Identifier-Token ('N' für noun, 'V' für verb, 'ADJ' für adjective, etc.). Je nach Wortart liegen dann weitere Informationen vor. Bei Substantiven wird das Geschlecht und die entsprechende Deklination, bei Verben die entsprechende Konjugation und der Deponentien-Identifier ('DEP') angegeben, sollte er anfallen.

Nach all diesen Informationen wird mit weiteren Whitespaces zu 102 Charakteren aufgefüllt. Nun Folgen eine Reihe Flags (laut der Website des Wörterbuchs geben diese Flags Informationen über Herkunft, Häufigkeit, etc. eines Eintrags an) und darauf schließlich die Übersetzungen. Diese letzten Informationen sind für die Flexion und demnach für das Programm allerdings irrelevant.

Bevor der nächste Schritt durchgeführt werden kann, werden die Einträge noch einmal etwas formatiert, Majuskel werden in Minuskel umgeformt und jegliche 'j' werden durch 'i' ersetzt.

### Anmerkung:

Das für diese Arbeit benutzte Wörterbuch ist dem Übersetzungsprogramm Words by William Whitaker entnommen, welches zur *freien* Benutzung freigegeben ist. Dieses ist in der englischen Sprache verfasst und enthält für über 39.000 Einträge auch Metainformationen und Übersetzungen. Es ist im Verzeichnis des Quellcodes als DICTPAGE.RAW auffindbar.

## 3.3 Pipelining

Da nun die Wortart eines Eintrags bekannt ist, kann dieser an die Pipelining-Methoden übergeben werden. Zweck dieser ist es, die vorliegenden Informationen so aufzubereiten, dass sie an die Flexionsmethoden übergeben werden können. Im Detail bedeutet das für:

- Substantive: Erkennen der Deklination und Übergeben an die passende Methode.
- Adjektive: Erkennen nach welchem Bildungsschema flektiert werden muss (erste & zweite oder dritte Deklination) und Übergabe an die passende Methode. Wenn Formen für den Komparativ und/oder den Superlativ angegeben sind, werden auch diese an ihre Methoden übergeben.
- Verben: Übergeben an die Verb-Flexions-Methode. Ist ein Verb als deponent markiert, wird an die gesonderte Deponentien-Flexions-Methode übergeben.
- Sonstige Wortarten: Die übrigen Worte werden entweder gar nicht flektiert oder enthalten ihre Formen direkt in der Zeile (Adverbien können zum Beispiel mit einer oder mit drei Formen angegeben sein).

## 3.4 Flexion

Durch das Pipelining sind die zu flektierenden Einträge nun auf die Flektionsmethoden angepasst. Zuerst wird der Stamm gebildet, an den dann die unterschiedlichen Endungen angehängt werden. Innerhalb der Methode werden durch einfache if-Anweisungen zusätzliche Suffixe zum Anhängen ausgewählt oder ausgeschlossen. Die resultierenden Strings werden in eine Liste eingefügt, die den Rückgabewert der Methoden darstellt.

## 3.5 Nachträgliches Einfügen von Sonderformen

Irreguläre Verben, Pronomina und weitere werden nicht nach den Schemata der Konjugationen/Deklinationen gebildet und müssen von Hand nachgetragen werden. Dies geschieht denkbar einfach: Eine Textdatei, die alle Formen hält, wird vom Programm zeilenweise eingelesen. Diese Textdatei ist wie folgt aufgebaut:

- Zeilen mit '#' als Präfix markieren einen neuen Eintrag. Das Wort in dieser Zeile wird dann als Grundform eingetragen.

- Zeilen mit '$' als Präfix markieren das Partizip eines irregulär konjugierten Verbs. Diese werden wie Adjektive der ersten beiden Deklinationen flektiert. Es müssen aber nicht alle Formen separat in die Textdatei eingetragen werden, es reicht aus, das Programm die entsprechende Flexionsmethode auf eine Grundform anwenden zu lassen.

- Zeilen mit '§' als Präfix markieren die Wortart.

- Zeilen ohne Präfix markieren einen normalen Eintrag.

## 3.6 Eintragen in die Hashmap

Die in den Schritten drei und vier gebildeten Formen werden nun verarbeitet. Für jede Grundform wird ein neues Objekt *WordEntry* angelegt. Für jede Form, die sich aus der Grundform ableiten lässt, wird ein Objekt *WordLink* erstellt, welches die Form selbst als String und einen 'Zeiger' auf das *WordEntry* Objekt, dem es entstammt, enthält. Die *WordLink*-Objekte werden dann nacheinander in eine Hashmap eingefügt. Die *WordEntry*-Objekte werden zusätzlich nach Wortart getrennt in Listen eingefügt, damit sie später sortiert werden können.

## 3.7 Bekanntheit der Worte prüfen

In diesem Schritt wird der zu analysierende Text zeilenweise eingelesen. Für jedes Wort wird dann überprüft, ob in der Hashmap ein entsprechendes *WordLink*-Objekt existiert. Sollte dies der Fall sein, wird bei jedem *WordEntry*-Objekt, das als mögliche Grundform eingetragen ist, die Zählvariable inkrementiert.

Worte die zuerst nicht erkannt werden, werden dann auf Prä- und Suffixe untersucht. Sollte ein Wort mit einem bekannten Präfix anfangen oder Suffix aufhören, werden diese entfernt und das geänderte Wort wird noch einmal in der Hashmap gesucht. Worte die endgültig nicht erkannt werden können, werden in eine Liste eingetragen und im letzten Schritt gesondert ausgegeben.

## 3.8 Worte nach Bekanntheit sortieren

Die nach Wortart getrennten Listen werden nun nach der Anzahl ihrer Auftritte sortiert. Der hierfür verwendete Sortieralgorithmus ist ein Treesort. Ein binärer Baum ist eigentlich nichts anderes als eine verkettete Liste, die zwei (anstelle von nur einem) Nachfolgerknoten kennt; einen linken und einen rechten. Beim Einfügen in den Baum werden die Elemente als linker Nachfolger eingetragen wenn sie öfter, als rechter wenn sie weniger oft vorgekommen sind. Sollte ein anderes Element bereits als Nachfolger eingetragen sein, wird das Einzufügende als Nachfolger dessen eingetragen.

Sobald alle Elemente eingetragen sind, kann der Baum ausgegeben werden. Dies geschieht rekursiv: Für jeden Knoten wird zuerst der linke Nachfolger, dann der Knoten selbst, dann der rechte Nachfolger ausgegeben. Das Element mit den meisten Auftritten wird also zuerst, das mit den wenigsten zuletzt ausgegeben.

## 3.9 Ergebnisse in eine Textdatei schreiben

In diesem letzten Schritt werden die Ergebnisse der Analyse kompiliert und in eine Textdatei geschrieben. Diese ist wie folgt aufgebaut:

1. Substantive, sortiert.
2. Adjektive, sortiert.
3. Verben, sortiert.
4. Worte die nicht vom Programm erkannt wurden (hier können auch Dopplungen auftreten).
5. Statistiken.

## 3.10 Benutzung des Programms

Führt man die Main-Methode der Klasse Main im Paket Main aus, wird man gebeten, einen Dateinamen anzugeben. Das Programm sucht dann im Ordner *src/texts* nach dieser Datei und führt sowohl die Textaufbereitung als auch die Analyse aus. Die Ergebnisse werden in einer neuen Datei, die sich im selben Ordner befindet, abgelegt.

Anstelle eines Dateinamens kann man auch 'default' angeben, dann wird die Analyse für die Texte von Cicero, Tacitus und Catull ausgeführt.

# Kapitel 4

# Probleme und ihre Lösungen

In diesem Kapitel soll es darum gehen, ein Licht auf die Probleme zu scheinen, die sich vor und während der Bearbeitung ergaben und wie sie überwältigt wurden.

**Erlangen eines Wörterbuchs** Das Erlangen eines Wörterbuchs steht am Anfang einer jeden Wortschatzanalyse. Wörterbücher gibt es viele und auch die bekannteren von Duden oder Langenscheidt sind mittlerweile elektronisch im Internet verfügbar. Da die Aufgabe in dieser Arbeit Worterkennung ist und nicht etwa eine Übersetzung, muss das verwendete Wörterbuch also speziellen Anforderungen genügen: Entweder sind zu jedem Wort Metainformationen (z.B. Wortart, Geschlecht oder Deklination/Konjugation) vorhanden, sodass sich alle möglichen Formen maschinell bilden lassen, oder aber es sind alle Flexionen bereits eingetragen. Die oben erwähnten Klassiker führen diese Informationen natürlich auf, das Online-Wörterbuch dict.cc hingegen zum Beispiel nicht; hier sind nur Übersetzungen eingetragen.

Entgegen aller Erwartungen stellte sich heraus, dass die großen Verleger nicht daran interessiert sind, Auszüge aus ihren Wörterbüchern bereitzustellen; nicht einmal für die Wissenschaft. Schließlich fand sich dann doch ein Wörterbuch, dass zumindest die Metainformationen enthält und für die FREIE Benutzung freigegeben ist: *Words by William Whitaker*. Dieses ist ein komplettes Wörterbuchprogramm, dessen zu Grunde liegende Textdatei, die das eigentliche Wörterbuch darstellt, für diese Arbeit verwendet wurde.

Weitere Informationen gibt es unter Words by William Whitaker.

**Textaufbereitung** Satz- und Sonderzeichen sowie Leerzeichen und Zeilenumbrüche stehen der Analyse im Wege. Bevor der Text analysiert werden kann, müssen diese also entfernt werden.

Hierfür wurden reguläre Ausdrücke benutzt, Schritt für Schritt wird dieser Prozess im Kapitel PROGRAMMABLAUF erläutert.

**Case-Sensitivity** Zur Verbesserung der Leserlichkeit werden heutzutage in lateinischen Texten Satzanfänge und Eigennamen großgeschrieben. Gerade bei den Satzanfängen ergeben sich dadurch eine Reihe Inkonsistenzen. Ein simples Ersetzen der Majuskeln durch Minuskel löst dieses Problem.

**'J' versus 'I'** Wie sich herausstellte, sind einige Worte im Wörterbuch mit 'J' eingetragen, die in den Texten mit einem 'I' geschrieben sind (zum Beispiel *jacere* anstelle von *iacere*). Obwohl unser deutsches Alphabet heutzutage aus lateinischen Buchstaben besteht und auch das 'J' enthält, ist eine Unterscheidung zwischen den Buchstaben 'J' und 'I' wohl erst im Mittelalter entstanden. Somit taucht der Buchstabe 'J' also in Texten der Antike nicht auf.

Um dieses Problem zu lösen reicht es, alle 'J' durch 'I' zu ersetzen, sowohl in den Texten als auch im Wörterbuch.

**Stringvergleiche** Stringvergleiche in Java sind eine heikle Angelegenheit. Strings gehören nicht zu den primitiven Datentypen, sodass Anstelle des ==-Operators die Methode String.compareTo(String anotherString) benutzt werden sollte, die einen lexikalischen Vergleich durchführt.

Beim Testen der Flexions-Methoden machte sich dieses Problem noch nicht bemerkbar, da diese hier noch mit internen Strings aufgerufen wurden und der ==-Operator das erwartete Ergebnis lieferte. Als die Flexions-Methoden aber beim Parsen des Wörterbuchs aufgerufen wurden und die entsprechenden Strings aus Variablen gelesen wurden, lieferte der ==-Operator nicht mehr das gewünschte Ergebnis.

Wie schon beschrieben löst sich das Problem durch das Verwenden der compareTo()-Methode. Für oft auftretende Token wurde Gebrauch von Enumerationen gemacht, sodass für jeden Eintrag im Wörterbuch nur ein einziges Mal ein aufwändiger lexikalischer Vergleich durchgeführt werden muss.

**Auswahl der passenden Datenstruktur** Die über 1.300.000 Formen, mit denen die Worte aus dem Text verglichen werden, müssen natürlich in einer Datenstruktur gespeichert werden. Eine verkettete Liste stellt den naivsten Ansatz dar, fällt aus Gründen der Effizienz jedoch weg. Um festzustellen, dass ein Wort nicht bekannt ist, muss dieses nämlich mit jedem Eintrag verglichen werden.

Eine Idee zur Verbesserung der Effizienz ist ein Filtern der Einträge nach ihren Anfangsbuchstaben. Man legt also 26 Listen an (oder ein Vielfaches davon), in denen jeweils nur Worte mit demselben (denselben) Anfangsbuchstaben einsortiert werden. Obwohl sich auf diese Art und Weise ein Großteil aller Worte noch vor dem Durchsuchen ausschließen lassen, ist auch dieser Ansatz nicht der Beste; die Buchstaben in der lateinischen Sprache sind natürlich nicht gleichmäßig verteilt.

Die Datenstruktur, welche den Ansprüchen gerecht wird, ist die Hashmap. Diese benutzt eine Hashfunktion um Einträge gleichmäßig in einem Array zu verteilen. Elemente lassen sich mit konstanter Geschwindigkeit sowohl einfügen als auch auslesen (vorrausgesetzt das Array ist nicht überfüllt).

**Irreguläre Verben** Die irregulär Konjugierten Verben lassen sich nicht an Hand eines festen Schemas beugen. Diese müssen also von Hand nachgetragen werden. Der technische Aspekt hierzu wird im Kapitel Programmablauf erläutert.

**Pronomina** Ähnlich wie die irregulären Verben werden Pronomina wie *qui, quae, quod* oder *is, ea, id* nicht nach den Schemata der fünf Deklinationen gebildet. Diese werden auch von Hand nachgetragen.

**Pipelining** Für jede Wortart gibt es eine eigene Methode für das Pipelining. Die Wortart allein reicht allerdings noch nicht aus, um den Aufbau eines Zeileneintrags zu erfassen. Im Wörterbuch werden je nach bekannten Formen unterschiedlich viele Grundformen pro Eintrag angegeben.

Beispielhaft: Das switch-Statement, das die Adjektive anhand der Anzahl der angegebenen Formen einteilt, kennt acht unterschiedliche Fälle. Da aus der Anzahl allein immer noch nicht bekannt ist, welche Formen überhaupt vorliegen, müssen danach weitere Vergleichsoperationen durchgeführt werden. Erst dann können die korrekten Flexions-Methoden ausgewählt und mit den korrekten Parametern aufgerufen werden.

**Präfixe und Suffixe** Aufgrund der synthetischen Wortbildung der lateinischen Sprache gibt es die Möglichkeit Worte durch Prä- und Suffixe zu erweitern. Dies findet hauptsächlich bei Verben Gebrauch (*ad* & *ire* $\rightarrow$ *adire*), wird aber auch bei anderen Wortarten verwendet (*senatoris* & *-que* $\rightarrow$ *senatorisque*).

Dieses Problem wird wenig elegant gelöst, indem jedes nicht erkannte Wort auf Prä- und Suffixe überprüft wird und dann, nach einem Entfernen dieser, noch einmal gesucht wird (es wird also ein Brute-Force durchgeführt).

# Kapitel 5

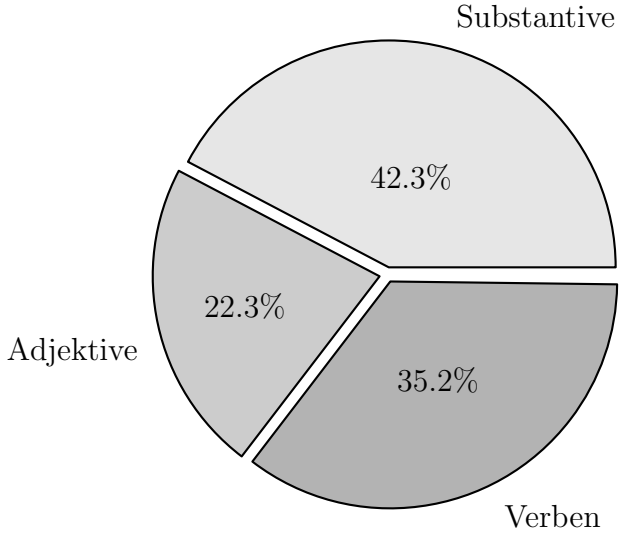
# Ergebnisse

Endlich sind alle notwendigen Schritte ausgeführt und es kann auf die Ergebnisse der Analyse eingegangen werden.

## 5.1 Gespräche in Tusculum

Die *Tusculanae disputationes* oder Gespräche in Tusculum sind eine Sammlung von fünf Büchern, in denen der Autor Marcus Tullius Cicero über Leben und Tod philosophiert.

Das Programm zählt insgesamt 47.422 Worte, von denen 2.212 Worte dem Programm nicht bekannt waren. Außerdem wurden 1.991 unterschiedliche Substantive, 1.052 unterschiedliche Adjektive und 1.657 unterschiedliche Verben gezählt.



Das am häufigsten auftretende Substantiv ist *animus* (Geist, Verstand, Seele) und wurde 409 mal gezählt. Weiterhin fallen *malum/malus* (Übel, Leid; 247, 242), *dolor* (Schmerz; 187), *aegritudo* (Unwohl, Kummer; 164) und *mors* (Tod; 130), sowie *vita* (Leben; 186) und *bonum/ bonus* (das Gute, Sittlichkeit aber auch Glück oder Wohl; 180, 162) auf.

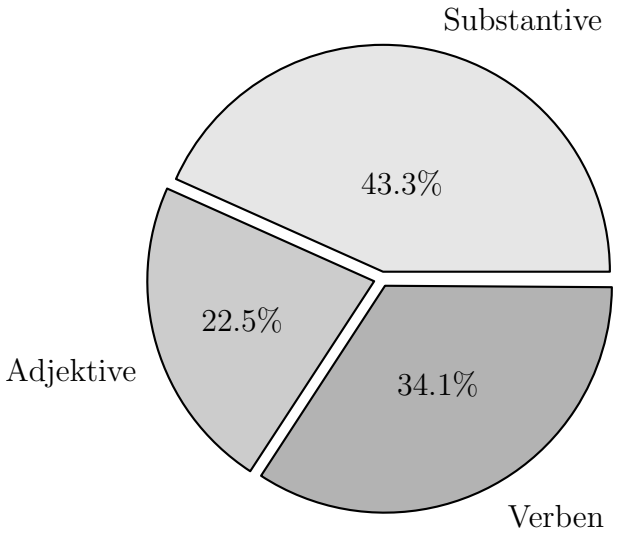
Bei den Adjektiven sieht es ähnlich aus, *bonus* (258) und *malus* (161) stehen wieder weit oben. Weiterhin tauchen *verus* (wahr; 193), *beatus* (glücklich, 142) und *sapiens* (weise; 123) auf.

Das meistbenutzte Verb ist *dico* (ich sage), welches 479 mal auftritt. Sonst lässt sich unter den meistbenutzten Verben nichts Besonderes feststellen.

## 5.2 Historien

Die *Historiae* oder Historien wurden von Publius Cornelius Tacitus verfasst und sind das erste von zwei Geschichtswerken dessen. Die Historien beschreiben die politischen und militärischen Geschehen von 69 nach Christus, das Jahr in dem Kaiser Nero starb, bis 96 nach Christus.

Das Programm zählt insgesamt 51.496 Worte, von denen 3.751 Worte dem Programm nicht bekannt waren. Außerdem wurden 2.330 unterschiedliche Substantive, 1.214 unterschiedliche Adjektive und 1.837 unterschiedliche Verben gezählt.



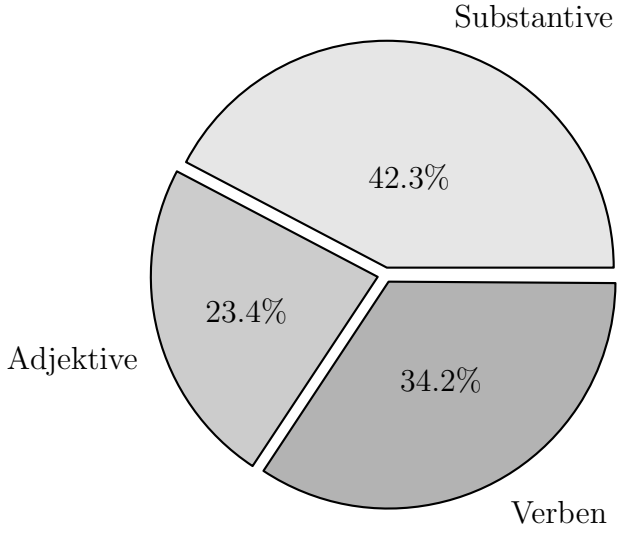
Die Häufigkeitsliste der Substantive wird angeführt von *miles* (Soldat; 314). Es folgen *legio* (Legion; 311), *bellum* (Krieg; 282), *exercitus* (Heer; 220), *dux* (Führer oder Feldherr; 163) und *armum* (Waffe; 151).

Bei den Adjektiven und den Verben lassen sich keine Sonderheiten feststellen.

## 5.3 Carmina

Die *Carmina* sind eine Sammlung von 116 Gedichten über verschiedenste Themen, die der Dichter Gaius Valerius Catullus in seinem Leben verfasste.

Das Programm zählt insgesamt 13.245 Worte, von denen 1.175 Worte dem Programm nicht bekannt waren. Außerdem wurden 1.464 unterschiedliche Substantive, 809 unterschiedliche Adjektive und 1.183 unterschiedliche Verben gezählt.




An Substantiven fallen *amor, amos* (Liebe; 53, 44), *puella* (Mädchen; 42), *mater* (Mutter; 30) ins Auge.

Auch hier bieten die Adjektive und die Verben keine interessanten Informationen.

## 5.4 Fazit

Es scheint, dass einzig die Substantive die Inhalte der Texte widerspiegeln. Vor allem bei den Verben sind die am häufigsten auftretenden Worte immer die Selben (z.B. sprechen, sehen, gehen, sein). Außerdem ist es ohne eine syntaktische Analyse unmöglich, eine akkurate Zählung durchzuführen. Ein Beispiel für diese Problematik bietet das Wort *bellum*, das entweder dem Substantiv *bellum, bello* (Krieg) oder dem Adjektiv *bellus, -a, -um* (hübsch) entspringen kann.

Weiterhin liefert die Analyse der relativen Häufigkeiten zwischen den Wortarten ein ziemlich ernüchterndes Ergebnis, auch wenn durch obiges Problem verzerrte Angaben entstehen.

Selbst wenn das Programm im jetzigen Zustand noch nicht mächtig genug ist wertvolle Informationen zu liefern, legt es den Grundstein für eine Reihe von Erweiterungen und Verbesserungen, die es hoffentlich eines Tages zu einem nützlichen Werkzeug weiterentwickeln. Ideen diesbezüglich werden im nächsten, letzten Kapitel erläutert.

# Kapitel 6

# Erweiterungsmöglichkeiten

In diesem abschließenden Kapitel werden ein paar Möglichkeiten vorgestellt, das Programm zu erweitern und zu verbessern.

## 6.1 Verbesserungen

Vorab ist anzumerken, dass das Programm in seiner aktuellen Form nicht perfekt ist. Es gibt hier und da Kleinigkeiten, die man noch verbessern kann. Bevor also funktionelle Erweiterungsmöglichkeiten diskutiert werden, sollen hier ein paar Optionen erwähnt werden, die man zuerst abarbeiten kann (diese Kleinigkeiten bieten durchaus eine gute Möglichkeit, sich mit dem vorhandenen Code auseinanderzusetzen und diesen zu durchdringen).

### Griechische Worte

Es sind im Wörterbuch auch griechische Substantive vermerkt, für die kein Deklinationstoken vorhanden ist; dennoch können diese gebeugt werden (im Internet lassen sich für einzelne Worte Tabellen mit den entsprechenden Formen finden). Es muss also ein oder mehrere Schemata geben, die man in Code umsetzen kann.

### Inserts

Es treten im Wörterbuch einige Zeilen auf, die keinen vollständigen Eintrag darstellen, sondern nur eine einzelne Sonderform zu einer Grundform, die anderswo vermerkt ist, anzugeben (siehe *bobus* → *bos, bovis*). Man könnte diese natürlich nachträglich von Hand einfügen, eleganter ist es jedoch, das Programm die Grundform, welche (im Wörterbuch) auf der rechten Seite in der Übersetzung vermerkt ist, selbst erkennen zu lassen.

### User Interface

Ein kleines User Interface, das die Auswahl des zu analysierenden Textes mit einem Filebrowser ermöglicht, erleichtert die Benutzung des Programms ungemein. Weiterhin könnte man eine Funktion einfügen, sich die Flexionstabellen zu einzelnen Worten bilden zu lassen.

## 6.2 Funktionelle Erweiterungen

Es folgen nun die Erweiterungsmöglichkeiten, für die man tiefer in den Code schauen und diesen wohl auch substanziell verändern muss.

### 6.2.1 Hinzufügen grammatikalischer Aspekte

Aktuell werden beim Flektieren nur die unterschiedlichen Formen als Strings gebildet und in eine Liste eingefügt, welche den Rückgabewert der jeweiligen Flexionsmethode darstellt. Demnach gehen Informationen, beispielsweise bei Substantiven über Kasus, Numerus und Genus, verloren. Es bietet sich also an, die Klasse *WordLink* um eine Variable, die eben diese Informationen hält, zu erweitern. Damit sollte der Grundstein für eine Syntaktische Analyse gelegt sein.

### 6.2.2 Syntaktische Analyse

Die lateinische Sprache ist schon sehr alt und hat sich im Laufe der Zeit stark gewandelt. Eine syntaktische Analyse von Texten aus unterschiedlichen Epochen ist sicherlich interessant, wenn auch eher für Linguisten als für Informatiker. Trotzdem ist die Voraussetzung für eine umfangreiche Analyse die Hilfe einer Maschine.

**Tipp:**

Syntaktische Analysen sind der Informatik nicht fremd; ein Compiler muss eine solche durchführen. Lesen eines Buches oder das Besuchen einer Vorlesung über Compilerbau sind sicherlich gute Optionen, sich mit dem Prozess einer syntaktischen Analyse vertraut zu machen.

### 6.2.3 Semantische Analyse oder Übersetzung

Fern in der Zukunft liegt eine semantische Analyse oder eine Übersetzung. Beide setzen voraus, dass das Programm die Worte eines Satzes syntaktisch

korrekt erfasst und Verbindungen zwischen diesen herstellt. Dies ist allerdings ein gigantisches Unterfangen für eine studentische Arbeit; selbst der Übersetzer von Google tut sich mit längeren Übersetzungen schwer.