Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Automatisierte Strukturierung von Benennungslisten

Studienarbeit Sprach- und Globalisierungsmanagement

Jan Oevermann (46594)
Hochschule Karlsruhe - Technik und Wirtschaft

3. Juli 2014

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangssituation

Nach einer manuellen oder maschinellen Termextraktion beginnt oft der mühseligste Teil der deskriptiven Terminologiearbeit: das Bereinigen und Strukturieren der aus dem Text gewonnenen Benennungsliste.[1] Da diese bei großen Textkorpora oft sehr umfangreich ist, stellt eine Recherche fast immer einen großer Zeitaufwand dar. Die begriffliche Ordnung selbst kann zwar (noch) nicht automatisiert werden jedoch die davor stattfindende Strukturierung der Benennungsliste.

Abbildung 1.1: Arbeitsablauf nach der Termextraktion

Ansatzpunkt für eine solche automatisierte Strukturierung können Webquellen wie linguistische Thesauri, Wortlisten oder Enzyklopädien sein, die das in ihnen gesammelte Wissen frei zur Verfügung stellen. Bekanntestes Beispiel dafür ist die Online-Enzyklopädie »Wikipedia«, die über eine Programmierschnittstelle (API) angesprochen werden kann. Aber auch andere Projekte aus dem FLOSS[2]-Bereich stellen teilweise in Handarbeit gepflegte Listen und Datenbanken unter freien Lizenzen bereit.

Bisher fehlt es an Werkzeugen, dieses umfangreiche (Sprach-)Wissen mit der Problematik der Benennungsstrukturierung zu verknüpfen, um Terminologen bei der begrifflichen Ordnung von Termkandidaten zu unterstützen.

[1] Die Bezeichnungen *Benennung*, *Wort* und *Term* werden in dieser Arbeit synonym verwendet
[2] Free/Libre Open Source Software

1.2 Zielsetzung

Ziel der Projektarbeit ist die konzeptionelle Ausarbeitung von Vorgehensweisen zur automatisierten Strukturierung von Benennungslisten mit Hilfe verschiedener programmatischer Methoden und dem Einbinden externer (Web-)Quellen. Zusätzlich soll eine prototypische Implementierung des Konzepts als Webanwendung vorgenommen werden, um die verwendeten Methoden und Annahmen zu testen.

Neben statistischen Methoden aus der Computerlinguistik soll auch eine Kompositumszerlegung, sowie das Erkennnen von Synonymen und anderen semantischen Relationen implementiert werden.

Die Ergebnisliste soll die eingegebenen Termkandidaten in bestimmte Benennungsgruppen (im besten Fall Begriffsgruppen) einteilen und ggf. spezielle Beziehungstypen kennzeichnen (Synonyme, Kategorien, etc.)

1.3 Vorgehensweise

Als Grundlage für die weitere Vorgehensweise dienen die Ergebnislisten einer maschinellen[3] Termextraktion des Wikipedia-Artikels »smart fortwo«[4] (ca. 700 Benennungen) sowie einer manuellen[5] Termextraktion der offiziellen »smart fortwo Betriebsanleitung« (ca. 220 Benennungen).

In einem ersten Schritt soll ermittelt werden, inwieweit eine Lemmatisierung der entstandenen Termkandidaten nötig ist und wie diese erfolgen kann. Dazu werden im Speziellen der Abgleich mit einer umfassenden Wortliste und programmatische Methoden untersucht.

Anschließend sollen zu einer ersten Beziehungsbildung statistische Verfahren aus der Computerlinguistik angewandt werden. Zur morpholgischen Beziehungsfindung soll eine Kompositumszerlegung implementiert werden. Dazu sollen verschiedene Methoden untersucht und ggf. kombiniert werden. Hierbei soll auch geprüft werden, ob bestehende Webservices oder Datenbestände genutzt werden können.

In einem weiteren Schritt soll die Einbindung eines externen linguistischen Thesaurus (oder desses Datenbestand) zur Synonymfindung geprüft werden. Als weitere semantische Relation sollen mögliche Abstraktionsbeziehungen durch das Einbinden der Wikipedia-API untersucht werden (im Speziellen der Kategorie-Systematik von Wikipedia).

[3] Das Werkzeug zur maschinellen Termextraktion *stex* (Simple Term Extractor) wurde selbst entwickelt und liegt der Arbeit bei. Die Termkandidaten setzen sich hauptsächlich aus Substantiven zusammen.
[4] http://de.wikipedia.org/wiki/Smart_Fortwo (Version vom 1. Juni 2014)
[5] Die manuelle Termextraktion wurde auf Basis des Index erstellt.

Für alle Schritte sollen verschiedene Beziehungsarten und ihre jeweiligen Bewertungen sowie ein Rahmen für die Einzelbewertung von Benennungen (unabhängig von einer konkreten Beziehung) konzipiert werden. Basierend auf diesen Bewertungen soll ein Vorgehen zur Strukturierung der Liste abgeleitet werden. Des weiteren soll auch die Möglichkeit der Kennzeichnung bestimmter Beziehungstypen entworfen werden.

Im letzten Schritt der Arbeit wird die Darstellung der Ergebnisse aus den vorherigen Kapiteln betrachtet und prototypisch umgesetzt. Dabei sollen durch Kombination der entwickelten Methoden eine strukturierte Liste von Termkandidaten entstehen, die wesentliche Vorteile gegenüber der unstrukturierten Variante bietet.

2 Statistische Wortähnlichkeit

Besonders im Bereich der Wortähnlichkeitsanalyse erzielen rein statistische Verfahren gute Ergebnisse mit vergleichbar wenig Rechen- und Implementierungsaufwand (vgl. Carstensen 2010:131). Im Gegensatz zu linguistischen Verfahren, die auf morphologischer oder lexikalischer Basis logische Bestandteile des Wortes untersuchen, werten statistische Verfahren *lediglich* Zeichenketten aus, ohne dabei etwaige Wort- oder Morphemgrenzen zu berücksichtigen. Durch diese Vorgehensweise können selbst große Textkorpora schnell und effizient untersucht werden.

2.1 Sørensen-Dice-Koeffizient

Eine weitverbreitete Metrik, die die Ähnlichkeit zweier Terme wiedergibt ist der *Sørensen-Dice-Koeffizent*[1] (im Folgenden nur *Dice-Koeffizient* nach Carstensen). Zur Berechnung dieses Wertes wird auf das sogenannte *N-Gramm-Modell* zurückgegriffen, bei dem aus einzelnen Termen alle vorkommenden Teilzeichenketten (inkl. Leerzeichen) einer Länge n extrahiert werden. Gängige Werte für n sind 1 (Unigramme), 2 (Bigramme) oder 3 (Trigramme) (vgl. Carstensen 2010:125). Beim Vergleich zweier Terme (a, b) wird über die Menge der N-Gramme der Terme (T) die Anzahl der gemeinsamen Vorkommen im Verhältnis zur Gesamtzahl berechnet:

$$Dice(a, b) = \frac{2 \cdot |T(a) \cap T(b)|}{|T(a)| + |T(b)}$$

Der Wert des Dice-Koeffizienten liegt dabei immer zwischen 0 und 1. Zur Veranschaulichung der Methode werden die Terme „Hase“ und „Nase“ über Trigramme ($n = 3$) miteinander verglichen (Leerzeichen werden als $\sqcup$ dargestellt):

$$Hase = {\sqcup\sqcup}h, {\sqcup}ha, has, ase, se{\sqcup}, e{\sqcup\sqcup}$$

$$Nase = {\sqcup\sqcup}n, {\sqcup}na, nas, ase, se{\sqcup}, e{\sqcup\sqcup}$$

Bei einer Gesamtzahl von 12 Trigrammen $(2 \cdot 6)$ weißen die beiden Terme drei Gemeinsamkeiten auf $(ase, se{\sqcup}, e{\sqcup\sqcup})$. Daraus errechnet sich ein Dice-Koeffizient von:

$$Dice(Hase, Nase) = \frac{2 \cdot |3|}{|12|} = 0,5$$

[1] Auch unter den folgenden Bezeichnungen bekannt: Dice-Koeffizent oder Sørensen-Index.

Die Methode hat nach Versuchen mit typischen Termlisten gute Ergebnisse erzielt. Durch die Berücksichtigung der Wortlänge (bzw. der Anzahl an N-Grammen) erzielen auch lange Benennungen gute Ähnlichkeitswerte (die auch auf eine semantische Verwandschaft schließen lassen):

$$Dice(\textit{Straßenverkehrsordnung, Straßenverkehr}) = 0,7$$

$$Dice(\textit{Bremseingriff, Bremseneingriff}) = 0,88$$

2.2 Levenshtein-Distanz

Als alternative Methode zur Berechnung von Term-Ähnlichkeiten kommt die *Levenshtein-Distanz*[2] in Frage (vgl. Carstensen 2010:558). Dabei werden die benötigten Änderungsschritte (Ersetzung, Einfügung, Löschung) gezählt, die nötig sind, um einen Term in einen anderen umzuwandeln. Bei den beiden Termen „Hase“ und „Nase“ ist das lediglich die Ersetzung von „H“ mit „N“ :

$$Levenshtein(Hase, Nase) = 1$$

Diese Methode wird vor allem in der automatischen Rechtschreibkorrektur und der unscharfen Suche verwendet. Beim Vergleich zweier Wörter, die sich ähnlich sind aber stark unterschiedliche Zeichenlängen haben kommt es allerdings zu schlechten Werten (hoher Wert = große Distanz):

$$Levenshtein(\textit{Straßenverkehrsordnung, Straßenverkehr}) = 7$$

Stärken zeigt die Methode allerdings beim Erkennen von starken Beziehungen zwischen Schreibungsvarianten (z.B. Diesel-motor $\leftrightarrow$ Dieselmotor), Tippfehlern (z.B. Coupé $\leftrightarrow$ Coupe) und nicht erkannten Flexionen (Designstudien $\leftrightarrow$ Designstudie). Alle im voherigen Satz genannten Beispiele haben eine Levenshtein-Distanz von 1 und eine dementsprechend hohe vermutete Beziehungsstärke (Zur Übertragung der Levenshtein-Distanz in das Bewertungskonzept der Anwendung siehe Abschnitt 5.2.1).

[2] Auch als *Minimal Edit Distance* oder *Levenshtein-Editierdistanz* bekannt.

3 Grundformen und Wortstämme

3.1 Lemmatisierung

Zur Normalisierung der eingegebenen Termliste sowie zum einheitlichen Vergleich gemeinsamer Wortstämme[1] müssen Terme und ihre Wortstämme auf ihre Grundform (nicht flektierte Form) gebracht werden. Dieser Vorgang wird als *Lemmatisierung* bezeichnet (vgl. Perera/Witte 2005:636), da hierbei das Wort auf sein *Lemma*[2] zurückgeführt wird, also die Form, unter der ein Begriff in einem Lexikon zu finden ist. In der Computerlinguistik ist bei deutschen Texten eine solche Lemmatisierung üblich[3] (vgl. Carstensen 2010:383) und nötig, da in natursprachlichen Texten Wörter oft in ihrer flektierten Form vorkommen.

3.1.1 Flexion im Deutschen

Als *Flexion* bezeichnet man die Veränderung bzw. Anpassung von Wörtern nach bestimmten grammatikalischen Kategorien und Regeln (vgl. Habermann/Diewald/Thurmair 2009:12). Flexion ist hierbei als Oberbegriff für die Kategorien *Konjugation*, *Deklination* und *Komparation* zu verstehen (ebd.). Anpassungen können Hinzufügungen, Veränderungen oder Ersetzungen sein. Für die Lemmatisierung besonders interessant ist hierbei der Bereich der synthetischen Flexion, der sich mit der Umformung des Grundwortes beschäftigt (vgl. Kluckhohn 2004). In dieser Arbeit werden nur Verben und Substantive sowie deren synthetische Flexion näher betrachtet.

Konjugation Verben können nach Person, Numerus, Tempus, Modus und Genus Verbi (aktiv/passiv) konjugiert werden (vgl. Habermann/Diewald/Thurmair 2009:13). Verben können regelmäßig oder unregelmäßig konjugiert werden. Dabei können Affixe als Wortbildungsmorpheme verwendet werden oder auch komplette Umformungen stattfinden (z.B.: sein → ich bin).

[1] *Wortstamm* wird in dieser Arbeit analog zum englischen *stem* (dt.: Stamm) verwendet und entspricht den Grundmorphemen anderer Literaur (vgl. Fleischer/Barz 2007:45)

[2] In Habermann/Diewald/Thurmair auch als *Nennform* bezeichnet.

[3] Im Gegensatz zu englischen Texten, bei denen mit den ursprünglichen Wortformen gearbeitet wird.

Deklination Substantive, Adjektive und Artikel können nach Genus, Numerus und Kasus dekliniert werden. Da Substantive ein festes Genus haben, können sie nur nach Numerus (Singular/Plural) und Kasus (Fall) dekliniert werden. Bei beiden Kategorien kann die Endung des Substantivs ergänzt werden (Kind → Kinder); bei bestimmten Pluralformen werden auch Vokale in Umlaute umgewandelt (Baum → Bäume).

Komparation Einige Adjektive und manche Adverbien können nach ihrer Steigerungsform kompariert werden. Da Adjektive und Adverbien in dieser Arbeit nicht als Termkandidaten betrachtet werden, ist diese Form der Flexion nicht für die Lemmatisierung relevant.

3.1.2 Listenabgleich

Grundlagen Auf Grund der teilweise sehr komplexen Flektierung der deutschen Sprache ist die Lemmatisierung von Termen im Deutschen nicht rein über einen regelbasierten Algorithmus zu lösen (vgl. Perera/Witte 2005:636). Darum haben sich in der Computerlinguistik verschiedene Verfahren zum Abgleich mit bestehenden Lemma- bzw. Vollformlisten etabliert (vgl. Hausser 2002:244ff). Hierbei handelt es sich um händisch oder teilmaschinell erzeugte Listen, die für ein Lemma alle möglichen Flexionsformen enthält.

Durch den Zugriff auf eine solche Liste reduziert sich der Lemmatisierungsaufwand auf eine Listensuche nach der gegebenen flektierten Wortform. Eine erfolgreiche Lemmatisierung kann mit dieser Methode aber nur dann erfolgen, wenn die Flektion bzw. das Lemma in der Liste gepflegt wurde. Neologismen können bei einem reinen Listenabgleich nicht erkannt werden (vgl. Hausser 2002:250).

Eine gut gepflegte und frei verfügbare Lemmatisierungsdateien findet man z.B. bei Naber. Die Datei umfasst ca. 431.00 Vollformen (vgl. Naber 2013) und stammt ursprünglich aus dem Morphy-Projekt (vgl. Lezius 2000). Das ursprüngliche Format der Datei wurde an die technischen Anforderungen angepasst und in eine JSON-Datei umgewandelt. Desweiteren wurde eine alternative Lemmatisierungsdatei erstellt bei der das eigentliche Lemma auch als eine der mögliche Flexionsarten (entspricht *Grundform*) mit aufgenommen ist und dementsprechend auch als Suchschlüssel verwendet werden kann. Diese Änderung erhöht die Quote korrekter Treffer bei einer schreibungsunabhängigen[4] Suche erweist sich jedoch bei der algorithmischen Wortstammsuche als nachteilig (siehe Abschnitt 3.2.5). Deshalb werden beide Varianten verwendet (siehe Tabelle 6.2).

[4]Die Bezeichnung *schreibungsunabhängig* entspricht dem englischen *case insensitive* und bezieht sich ausschließlich auf die Groß-/Kleinschreibung

Funktion Für die Lemmatisierung von Komposita und deren Wortstämme kann die gleiche Funktion verwendet werden, wenn dabei Groß/Klein-Varianten berücksichtigt werden. Hierzu werden nacheinander mit verschiedenen Varianten Suchanfragen an die Lemmatisierungsliste gestellt. Bei einem Treffer wird das entsprechende Lemma zurückgegeben, bei keinem Treffer wird die nächste Variante getestet. Mögliche Varianten sind hierbei: Originalschreibung, Großschreibung des ersten Buchstaben (*de facto* Substantivierung), Kleinschreibung des ersten Buchstaben.

3.1.3 Entfernen der s-Flexion

Grundlagen Mit der Bezeichnung *s-Flexion* wird der „Flexionstyp von Substantiven bezeichnet, der als einziges Flexionssuffix ein *-s* (und ausdrücklich auch nicht ein Suffix *-es*) hat“ (Eisenberg 2007:820). Die s-Flexion tritt als *Genitiv-s* bei der Deklination (vgl. Eisenberg 2007:370 ff.), sowie in der Pluralbildung (vgl. Habermann/Diewald/Thurmair 2009:24) von Substantiven auf. „Die Zahl der Substantive mit s-Flexion steigt gegenwärtig stark an“ (Eisenberg 2007:821).

Bei der Genitivbildung wird die s-Flexion immer angewendet bei Substantiven auf: *-en, -em, -el, -er, -ler, , -ner, -end, -chen, -lein, -ig, -ich* (vgl. Eisenberg 2007:370). Auffällig ist hierbei, dass der letzte Buchstabe aller Endungen ein Konsonant ist. Dieses Muster spiegelt sich auch bei der Genitivbildung von Fremdwörtern wider, die bei Endung auf einen Konsonanten ebenfalls die s-Flexion anwenden (vgl. Eisenberg 2007:371). Die Pluralbildung mit s-Flexion erfolgt vor allem bei Personennamen, Kurzwörtern und Entlehnungen aus dem Französischen oder Englischen (vgl. Eisenberg 2007:819 f.).

Als Wortbildungssuffix für Substantive kommt *-s* kaum vor. Sein Gebrauch ist „deutlich umgangssprachlich markiert“ und „in der Literatursprache nur schwach ausgeprägt“ (Fleischer/Barz 2007:167). Beispiele für Substantive die mit Suffix *-s* gebildet werden, sind: *Klecks, Zeugs, Flaps, Stups oder Pups* (vgl. Fleischer/Barz 2007:167 f.). In fachsprachlichen Textkorpora sind solche Wörter mit hoher Wahrscheinlichkeit nicht zu finden und können deshalb für diesen Anwendungsfall ignoriert werden. In der Literatur sind für die Wortbildung per Derivation von Substantiven, Adjektiven und Verben keine Suffix mit der Konstellation *Konsonant + s* zu finden (ohne die oben genannte Ausnahme).

Die wenigen nicht-umgangssprachlichen Ausnahmen bei Substantiven der deutschen Sprachen, wie etwa *Gans*, *Kurs* oder *Mars* werden hierbei zu Gunsten des Regelfalls vernachlässigt. Ein Workaround[5] für diese falsch positiven Treffer ist, die ursprüngliche Form als Wortstamm mit aufzunehmen, also: *Gans* → *Gans, *Gan*.

Eine Ausnahme bilden Adverben der deutschen Sprache mit den Wortbildungssuffixen *-ens, -dings, -lings, -mals*, und *-wärts* (vgl. Fleischer/Barz 2007:285 ff.) bei denen

[5]Umweg zur Vermeidung von bekanntem Fehlverhalten eines technischen Systems

die oben genannte Kombination aus *Konsonant + s* als letzte Buchstaben anzutreffen ist. Adverben, die mit diesen Suffixen gebildet werden, kommen allerdings kaum in technischen Texten vor. In Sonderfällen können fehlerhafte Lemmatisierungen mit dem oben beschriebenen Workaround abgefangen werden.

Funktion Aus den oben genannten Mustern lässt sich eine zuverlässige Regel ableiten, die besagt, das Substantive, die auf die Kombination *Konsonant (ohne s)*[6] + *s* enden, durch das Entfernen des *s* lemmatisiert werden können.

Dadurch können Einträge, die nicht in der Lemmatisierungsliste gefunden werden, und den oben beschriebenen Anforderungen genügen, trotzdem lemmatisiert werden. Gerade bei speziellen oder neuen Fachbegriffen (z.B. zwei *Thoraxairbags*), Markennamen (z.B. mehrere *Smarts*) sowie Entlehnungen aus anderen Sprachen (z.B. des *Bordcomputers*) zeigt diese Methode ihre Stärken.

3.2 Komposita

3.2.1 Komposition im Deutschen

Das Bilden von Komposita ist neben der Derivation die wichtigste Wortbildungsart im Deutschen. Bei der Komposition (also der Bildung eines Kompositums) werden freie Wortstämme, lexikalische Morpheme oder Konfixe zu einem neuen Wort miteinander verbunden (z.B Baum, Haus → Baumhaus) (vgl. Fleischer/Barz 2007:45). Diese Wortstämme werden innerhalb eines Kompositums als *Konstituenten* bezeichnet. In einigen Sonderfällen kann die Komposition auch nur mit einzelnen Buchstaben, Zahlen oder aus phraseologischen oder onymischen Wortgruppen gebildet werden (vgl. Fleischer/Barz 2007:45).

Die Bildung von Komposita hat in der Terminologie die größte praktische Bedeutung in der Benennungsbildung (vgl. Drewer/Ziegler 2011:177). Das Erkennen und Zerlegen von Komposita (Dekomposition) hatte dementsprechend großen Einfluss in der Entwicklung der Anwendung.

Typen Im Deutschen werden zwischen verschiedenen funktionalen Arten von Komposita unterschieden. Im folgenden werden drei der wichtigen Kompositumsarten des Deutschen kurz vorgestellt.

[6] In der technischen Umsetzung wird geprüft, ob keiner der folgenden Buchstaben der vorletzte ist: *a, e, i , o, u, ä, ö, ü, s*

Determinativkomposita Bei Determinativkomposita besteht zwischen den Konstituenten ein Über- oder Unterordnungsverhältnis, das heißt ein (Wort-)Teil determiniert den anderen Teil (bestimmt ihn näher). Im Deutschen gilt hierbei grundsätzlich, dass der erste Konstituent den zweiten determiniert (vgl. Donalies 2005:57). Determinativkomposita sind in der Regel endozentrisch, das heißt, der determinierte Teil kommt auch im Kompositum selbst vor (im Gegensatz zu Possessivkomposita, siehe Abschnitt 3.2.1).

Determinativkomposita bilden den größten Anteil deutscher Komposita (vgl. Donalies 2005:52 + 58). Wichtig für die Lemmatisierung (siehe Abschnit 3.1) von Determinativkomposita ist, dass der zweite Teil die grammatischen Merkmale und damit auch die Flexion des Kompositums festlegt (die sog. *Righthand Head Rule*) (vgl. Donalies 2005:54).

Kopulativkomposita Bei Kopulativkomposita stehen sich die Konstituenten gleichberechtigt gegenüber (z.B. bei *schwarz-weiß / schwarzweiß*). Im Gegensatz zu Determinativkomposita lassen sich die Wortteile in der Regel vertauschen ohne, dass sich der Sinn ändert (was auf eine Gleichberechtigung hinweist). Im Deutschen treten Kopulativkomposita fast ausschließlich bei Adjektiven auf (vgl. Donalies 2005:86).

In der Anwendung kann das falsch interpretierten Basis-Wortstämmen zur Folge haben (da hier automatisch der letzte Wortstamm ausgewählt wird). Da Kopulativkomposita im Allgemeinen und auch im speziellen Bereich der Fachsprachen aber eher selten vorkommen (vgl. Donalies 2005:84 f.) kann dies vernachlässigt werden.

Possessivkomposita Bei Possessivkomposita haben die Konstituenten ein determinatives, nicht kopulatives Verhältnis, bei dem der (näher) beschriebene Teil jedoch nicht im Wort selbst vorkommt (exozentrisches Kompositum) (vgl. Fleischer/Barz 2007:125). Dabei handelt es sich vorwiegend um Personen-, Pflanzen- und Tierbezeichnungen (vgl. Fleischer/Barz 2007:125). Für die technische Textkorpora sind hierbei nur die Possessivkomposita mit Numeral als Erstglied interessant, z.B. *Dreizylinder* → Motor mit drei Zylindern.

Für die Entwicklung der Anwendung ist hierbei zu beachten, das der externe Teil, nicht ohne weiteres hergeleitet werden kann. So kann *Dreizylinder* nur durch die Betrachtung der Wortstämme nicht zu *Motor* als Unterbegriff zugeordnet werden (sondern eher zu *Zylinder*, was semantisch nicht korrekt ist).

3.2.2 Analytische Zerlegung

Grundlagen Auf Grund der zahlreichen Möglichkeiten Komposita im Deutschen zu bilden, ist eine rein regelbasierte bzw. analytische Zerlegung ohne großen Aufwand nicht zu realisieren. Die Problematik lässt sich anhand eines Beispiels (vgl. Carstensen 2010:226) verdeutlichen:

> **Mögliche Zerlegungen von „Wählerstimmen“:**
> wähl [Vb-Stamm] + erst [Adj-Stamm] + imme [Nom-Stamm] + n [Pl]
> wähler [Vb-Stamm] + st [2. Per-Sg] + imme [Nom-Stamm] + n [Pl]
> wähler [Nom-Stamm] + stimme [Nom-Stamm] + n [Pl]

Die richtige Zerlegungsvariante kann nur über Auftretungshäufigkeiten in großen Textkorpora oder Wahrscheinlichkeitsbewertungen bestimmt werden (vgl. Carstensen 2010:226). Um eine solche Zerlegung umzusetzen, bedarf es also immer der Analyse großer Textmengen aus dem Fachbereich der zu analysierenden Termliste. Sie kommt deshalb nicht für die in dieser Arbeit behandelten Anwendung in Frage.

3.2.3 Bindestrichzerlegung

Grundlagen Eine Trennung von Kompositabestandteilen durch einen Bindestrich ist im Deutschen nur in einigen wenigen Fällen vorgesehen (vgl. Rat für deutsche Rechtschreibung 2006:45), kommt aber in der Praxis sehr häufig vor.

Legitime Anwendungen des Bindestrichs beschränken sich nach dem Regelwerk des Rat für deutsche Rechtschreibung auf die Zusammensetzung mit Abkürzungen, einzelnen Buchstaben und Ziffern (alle §§ 40 f.) sowie Eigennamen (§§ 46 & 48 ff.) und fremsprachlichen Entlehnungen (§ 45 E1 & E2) als auch zur Bildung von gleichrangigen Aneinanderreihungen (§ 43 f.). Des weiteren kann zur Hervorhebung oder besseren Leserlichkeit ein Bindestrich gesetzt werden (§ 45). In allen Fällen ist auf den Zusammenhalt von Sinneinheiten zu achten (z.B. *Flüssigwasserstoff-Tank* statt **Flüssigwasser-Stofftank*). Viele Redaktionsleitfäden geben auch den Einsatz eines Bindestrichs ab einer bestimmte Kompositumslänge vor (etwa ab vier Bestandteilen). Werden diese Regeln korrekt angewandt, kann eine Kompositumszerlegung an einer Bindestrichgrenze problemlos erfolgen.

Funktion Durch die vorher getroffenen Annahme kann eine einfache Suche nach Bindestrichen innerhalb eines Wortes und anschließender Auftrennung an diesen eine (Vor-)Zerlegung des Kompositums erfolgen. Dabei ist es unerheblich ob einer der vermuteten Wortstämme ein weiteres Kompositum darstellt.

Beispiele:
Common-Rail-Diesel → Common, Rail, Diesel (kompl. Zerlegung)
Tridion-Sicherheitszelle → Tridion, Sicherheitszelle (teilw. Zerlegung)

Nach einer Lemmatisierung der vermuteten Wortstämme wird auf jeden Treffer sowohl für das gesamte (vermutete) Kompositum ein Listenabgleich als auch eine algorithmische Wortstammsuche durchgeführt. Dort werden ggf. noch nicht zerlegte Komposita in ihre Stämme zerlegt.

3.2.4 Listenabgleich

Grundlagen Auf Grund der im Kapitel 3.2.2 beschriebenen Problematik bietet sich der Abgleich mit einer Liste an, die für ein Kompositum die jeweiligen Wortstämme zurückliefert. Eine Übersicht über die verschiedenen Organisationen, die sich mit den Erstellen solcher Listen befassen und den Nutzungsrechten, denen die Ergebnisse unterliegen hat DANTE E.V. zusammengestellt (vgl. DANTE E.V. 2014). Hierbei besonders hervorzuheben ist die Freie[7] Wortliste von LEMBERG, die rund 430.000 manuell gepflegte Wörter mit Worttrennungsinformationen enthält (vgl. LEMBERG 2014). Solche Listen werden hauptsächlich in Textverarbeitungsprogrammen für automatische Zeilenumbrüche verwendet. Da zusammengesetzte Wörter zunächst an ihren Kompositionsfugen getrennt werden sollten, enthält die Liste eine mehrstufige Trennung: Haupttrennstellen (mit = gekennzeichnet) sowie Nebentrennstellen 1. Ordnung (mit − gekennzeichnet) und 2. Ordnung (mit _ oder < gekennzeichnet) (vgl. KODYDEK 2001:3).

Beispiel: *Abfallentsorgungssystem*
Ab<fall=ent<sor-gungs=sy-stem

Funktion Da Haupttrennstellen immer an Wortfugen von Komposita auftreten (vgl. KODYDEK 2001:3) kann an ihnen die (oft) flektierte Form des Wortstamms abgelesen werden. Dadurch kann die Liste auch zur Kompositumszerlegung verwendet werden. Alle gefundenen Wortstämme werden anschließend lemmatisiert, um die Grundformen verschiedener (Teil-)Terme miteinander vergleichbar zu machen.

Für die Anwendung wurde die Liste in eine JSON-Datei umgewandelt, die das regulär geschrieben Kompositum als Suchschlüssel und (nur) die Wortstammgrenzen als Wert zurückgibt. Die Informationen zu Trennstellen 1. und 2. Ordnung wurden verworfen, da sie für die Kompositumszerlegung nicht benötigt werden.

[7] Der Ausdruck *Frei* wie in *Freie Software* bedeutet eigentlich *freiheitsgewährend*. LEMBERG hat für seine Liste die GPL (GNU General Public License) angestrebt, die diese Anforderugnen erfüllt.

3.2.5 Algorithmische Wortstammsuche

Grundlagen Im Bereich der Fachsprachen gibt es eine Tendenz zu sehr langen und teilweise neu kreierten Komposita. Diese werden oft nicht in allgemeinen Wortlisten (wie der verwendeten) gepflegt. Hiervon sind auch die im Deutschen beliebten Ad-hoc-Komposita betroffen. Um auch Komposita abzudecken, die nicht durch eine Bindestrichzerlegung oder einen Listenabgleich zerlegt werden können, wurde eine neue Vorgehensweise entwickelt: die algorithmische Wortstammsuche.

Funktion Die Idee des Vorgehensweise besteht darin, innerhalb eines Wortes für jede mögliche Zeichenkette einer Länge l ab einem Buchstaben n eine Suche gegen eine Lemmatisierungsliste durchzuführen. Bei Erfolg wird der entsprechende Teil des Strings entfernt und die Suche von neuem begonnen. Wird bei einem Durchlauf kein Treffer gefunden, wird der erste Buchstabe der verbleibenden Zeichenkette entfernt und von neuem begonnen.

Als Lemmatisierungsdatei wird dabei die Variante verwendet, bei der Grundformen nicht als Suchschlüssel auftreten. Damit wird der Tatsache entsprochen, dass Fugenelemente von Komposita ursprünglich aus Flexionssuffixen der Wortstämme entstanden sind (vgl. Fleischer/Barz 2007:136). Durch das Erkennen der flektierten Form, werden mögliche Fugenelemente mit getilgt (bei gleichzeitiger Rückführung des flektierten Wortstammes in seine Grundform).

Der Vorgang kann an einem Beispiel verdeutlicht werden (Mindestlänge $l = 3$):

> Vermutetes Kompositum: *Pannenset*
> Pan (n_0, l_3) = 0
> Pann (n_0, l_4) = 0
> Panne (n_0, l_5) = 0 (nicht erkannt, da Grundform)
> Pannen (n_0, l_6) = 1 -> erster Wortstamm: *panne*
> set (n_6, l_3) = 0 -> zweiter Wortstamm: *set* (Restwort mit Mindestlänge)

Im gezeigten Beispiel wurde zwar der letzte Wortstamm *Set* nicht erkannt, da es sich nicht um eine flektierte Form handelt, jedoch geht der Algorithmus automatisch von einem Wortstamm aus, wenn es sich um den größtmöglichen Wortrest mit Mindestlänge $l = 3$ handelt.

Weitere Beispiele für die der Algorithmus gut funktioniert, sind: *Tagesstrecke*, *Sonnenblende*, *Kindersitz* oder *Getränkehalter*.

4 Semantische Relationen

Neben der rein statistischen und morphologischen Betrachtung von Wortbeziehung, bilden semantischen Relationen zwischen Wörtern die wohl stärksten Verbindungen. Sie kommen in der Phase der Strukturierung der eigentlichen begrifflichen Ordnung am nächsten.

Typische semantische Relationen in der Sprache sind Bedeutungsverschiedenheit (*Polysemie*), Bedeutungsgleichheit (z.B. *Synonymie*), Bedeutungsgegensatz (*Antonymie*) und Bedeutungshierarchie (z.B. *Hyperonymie*) (vgl. Busse 2009:105). Für die Anwendung untersucht und implementiert werden die Beziehungsarten *Synonymie*, *Hyperonymie/Hyponymie* und der Grad der *Vernetzung* eines Wortes.

4.1 Synonymie

Nach Definition ist Synonymie die Bedeutungsgleichheit zweier Wörter, so dass sie in jedem Kontext substituiert (ausgetauscht) werden können. In der Literatur ist es umstritten ob eine solche strikte Bedeutungsgleichheit überhaupt geben kann (vgl. Busse 2009:104). Für die Entwicklung der Anwendung spielt dieser Punkt nur eine untergeordnete Rolle, da in der Strukturierung nur eine Vorgruppierung und ggf. Synonymkennzeichnung stattfindet. Die eigentliche begriffliche Ordnung und damit auch das Bewerten eines Synonymkandidaten bleibt einem Terminologen überlassen.

4.1.1 Listenabgleich

Grundlagen Mit der Zeit haben sich im Internet einige linguistischen[1] Thesauri als Formulierungshilfen etabliert. Erklärtes Ziel aller Projekten ist, dem Nutzer für ein gegebenes Wort bedeutungsähnliche Benennungen oder Synonyme zurückzuliefern. Hervorzuheben ist hierbei das Projekt *OpenThesaurus*[2] von Daniel Naber, das nach eigenen Angaben ein „deutschsprachiges Wörterbuch für Synonyme und Assoziationen“

[1] Im Gegensatz zu dokumentationswissenschaftlichen Thesauri, die sich mit verschiedenen Wortrelationen beschäftigen.

[2] Zu finden unter: http://www.openthesaurus.de

ist (siehe OpenThesaurus). Die dort gesammelten und verwendeten Datensätze stehen unter der Lizenz LGPL[3] frei zur Verfügung und werden als Download angeboten.

Die Datensätze von OpenThesaurus sind in der Form:

$$T_1 = T_2, T_3, ... T_n$$

aufgebaut, wobei T_1 nicht immer auch Synonym von T_n ist. Da aber (echte) Synonyme per Definition austauschbar sind, wurden die Datensätze entsprechend umgeformt:

$$\begin{aligned} T_1 &= T_2, T_3, ... T_n \\ T_2 &= T_1, T_3, ... T_n \\ T_3 &= T_1, T_2, ... T_n \\ &... \end{aligned}$$

Dadurch konnte die künstliche Erhöhung der Einträge die Trefferhäufigkeit verbessert werden. Auf Grund der nicht-technischen Quelle ist allerdings zu beachten, dass es sich bei den Treffern oft nicht um Volläquivalente handelt, sondern um oft nur um ähnliche Benennungen. Da Synonyme während der begrifflichen Ordnung von einem Terminoligen festgelegt werden, kann dies vernachlässigt werden. Innerhalb der Strukturierungsphase sind Gruppierungen ähnlicher Benennungen in der Regel zulässig und korrekt (so lange diese nicht fälschlicherweise als Synonyme gekennzeichnet werden).

Funktion Die Funktion zum Abfragen der Synonyme ist eine simple Listenschlüsselsuche, die bei einem Treffer, die entsprechenden Werte als Liste zurückliefert. Wird keine Eintrag in der Liste gefunden, wird eine leere Liste zurückgegeben (keine Synonyme). Die Listensuche erfolgt mit Berücksichtigung der Groß-/Kleinschreibung.

Cabriolet → Cabrio, offener Wagen, Kabrio, Kabriolett

4.1.2 Formulierungsmuster

Ein in dieser Arbeit nicht weiter untersuchter Ansatz ist, Artikel von Online-Enzyklopädien oder ähnliche Datenquellen nach Formulierungsmuster zu durchsuchen, die auf eine Synonymie hinweisen. Innerhalb eines Fließtextes werden Synonyme oft mit geläufigen Floskeln wie „*auch ... genannt*“, „*auch als ... bekannt*“ oder „*manchmal als ... bezeichnet*“ eingeführt.

Diese Methode kann in weiterführenden Arbeiten detaillierter betrachtet werden und die Trefferhäufigkeit bei der Synonymsuche unter Umständen erhöhen.

[3]GNU Lesser General Public License: kann in jeder (auch proprietärer) Software verwendet werden.

4.2 Kategorisierung

Online-Enzyklopädien wie Wikipedia[4] bieten mit ihrem frei zugänglichen Wissen nicht nur reine Lexikoneinträge sondern auch eine semantische Kategorisierung der Lemmata zum Einordnen in Themenbereiche (inhaltliche Systematik):

> „Kategorien sind in der Wikipedia ein Mittel, mit dem Seiten nach bestimmten Merkmalen eingeordnet werden können. Eine Seite kann einer oder mehreren Kategorien zugewiesen werden; die Kategorien können ihrerseits wieder anderen Kategorien zugeordnet sein (Hierarchisierung in Unter- und Oberkategorien)." (Wikimedia Foundation Inc. 2014b)

Diese beschriebene Hierarchisierung entspricht weitgehend der begrifflichen Ordnung in Ober- und Unterbegriffe (abstrahierende Begriffssysteme) aus der Terminologie und kann entsprechend von großem Nutzen bei der Strukturierung von Benennungen sein. Neben der eigentlichen Einordnung in Kategorien leistet das Wikipedia-Backend weitere Arbeit, wie z.B. die automatische Weiterleitung bei Flektionen (Thesauri → Thesaurus), Synonymen (Benzinmotor → Ottomotor), Abkürzungen (USA → Vereinigte Staaten), Alternativschreibungen und weiteren Grenzfällen. Handelt es sich um einen Fall von *Polysemie* (Mehrdeutigkeit), schaltet Wikipedia zunächst eine Seite zur *Begriffsklärung* zwischen, auf welcher der Nutzer den richtigen Begriff auswählen muss (z.B. Golf → Golf (Meer), Golf (Sport), VW Golf, etc.).

4.2.1 API

Wikipedia stellt über die von der Enzyklopädie verwendete Software MediaWiki eine API (Application Programming Interface) zur Verfügung, um auf die Artikel-Datenbank zuzugreifen. Diese API kann über URL-Aufrufe verschieden Informationen über Artikel oder diese selbst zurückliefern. Die Nutzung der Schnittstelle ist kostenlos, die Zugriffsfrequenz allerdings begrenzt.

Je nach Abfrageparamter werden dabei unterschiedliche Daten zurückgeliefert. MediaWiki erlaubt es, nur die Kategorien eines Artikels zurückzuliefern: die Funktion die für die Anwendung benötigt wird. Die verwendeten Parameter können im kommentierten Quellcode der Anwendung nachvollzogen werden. Bei der Abfrage werden automatische Weiterleitungen (siehe oben) zugelassen.

Eine Beispiel-URL für eine solche Abfrage kann lauten:
`https://de.wikipedia.org/w/api.php?action=query&prop=categories&format=json&titles=Chiptuning&redirects=true&callback=`

[4] Die deutschsprachige Wikipedia ist zu erreichen unter; http://de.wikipedia.org

4.2.2 Verarbeitung

Nach der Anfrage liefert die MediaWiki-API eine JSON[5]-Datei zurück, die anschließend von der Anwendung weiterverarbeitet wird. Hierbei wird zunächst eine Filterung von Kategorien vorgenommen, die intern bei Wikipedia verwendet werden (z.B. *Löschkandidat* oder *Qualitätssicherung*) und Seitentypen, die nicht automatisiert verarbeitet werden können (z.B: die schon erwähnten Begriffsbestimmungsseiten). Nach der Filterung bleibt eine Liste der Kategorien, in die eine Benennung eingeordnet ist. Beispiel:

Chiptuning → Fahrzeugtuning, Leistungssteigerung (Verbrennungsmotor)

4.3 Vernetzung

Nach Anwendung aller aufgeführten Verfahren entstehen zwischen den Termkandidaten Verbindungen unterschiedlicher Stärke.

4.3.1 Annahme

Eine Benennung, die viele Verbindungen mit einer hohen durchschnittlichen Verbindungsstärke besitzt, hat innerhalb des untersuchten Textkorpus eine besondere Rolle und daraus folgernd eine besondere Wichtigkeit. Deshalb fließt der Vernetzungsgrad eines Terms mit in die Benennungsbewertung ein.

4.3.2 Berechnung

Die Berechung basiert auf der Gesamtzahl an Wortverbindungen, die nicht 0 sind (n) und den jeweiligen Verbindungsstärken ($Beziehung_n$).

$$Vernetzung = 2 \cdot (\frac{Beziehung_1 + Beziehung_2 + \ldots + Beziehung_n}{n})^2$$

Die Faktorierung ($2x^2$) dient zur Einpassung in das Bewertungskonzept der Anwendung. Da der Vernetzungsgrad auf einer Annahme beruht und in manchen Fällen zu hohe Werte liefert, wird er in der Gesamtbewertung eher gering gewichtet.

[5]Technisch gesehen eine JSONP-Datei

5 Strukturierung

5.1 Benennungsbewertung

Neben den Beziehungsstärken die zwischen den verschiedenen Benennungen berechnet werden, müssen auch für die Benennungen einzeln Bewertungen vergeben werden. Diese Bewertungen sind wichtige Grundlage für die spätere Gruppierung. Durch die Bewertung einer Benennung werden sowohl Obergruppen als auch Gruppenzugehörigkiet bestimmt (siehe dazu auch Abschnitt 5.3.2).

5.1.1 Faktoren

Für die Benennungsbewertung werden verschiedene Faktoren mit unterschiedlicher Gewichtung verrechnet. Grundlage dafür sind bestimmte Annahmen, die im folgenden aufgeführt werden:

Benennung ist Oberbegriff (hohe Gewichtung) Diese Annahme lässt sich in zwei Fällen mit unterschiedlichen Wahrscheinlichkeiten treffen: Wenn ein Wort oder das Synonym eines Wortes die Kategorie eines anderen Wortes ist (hohe Wahrscheinlichkeit) und wenn ein Wort das Grundwort eines anderen Worts ist (mittlere bis hohe Wahrscheinlichkeit).

Benennung ist stark vernetzt (geringe Gewichtung) Diese Annahme lässt sich aus mehreren Faktoren herleiten: Ein Wort hat viele gemeinsame Wortstämme mit anderen Benennungen, ein Wort hat gemeinsame Kategorien mit anderen Wörten oder ein Wort ist Synonym eines anderen Wortes. Zusätzlich zählt dazu auch der in Abschnitt 4.3.2 beschriebene Vernetzungskoeffizient, der auch die statistischen Verfahren mit berücksichtigt.

5.1.2 Berechnung

Bei der Berechung des eigentlichen Benennungswertes werden einige Anpassungen vorgenommen, um auch sehr heterogene Benennungslisten verarbeiten zu können (z.B. unterschiedliche Präsenz bei Wikipedia oder in einer der Wortlisten).

Gemeinsame Wortstämme Bei der Berechnung des Wertes für gemeinsame Wortstämme (WS) mit anderen Benennungen wird die Anzahl der Wortstämme des Ausgangswortes berücksichtigt:

$$\text{Wortstämme}(a, b) = \frac{WS(a) \cap WS(b)}{WS(a)}$$

Kategorien Für Kategorien wird zunächst ein Gesamtwert für Kategorie-Faktoren berechnet, der dann anschließend durch die Anzahl der Kategorien des Ausgangswortes geteilt wird. Dieses Vorgehen ist nötig um Ungleichmäßigkeiten in der Wikipedia-Kategorisierung auszugleichen.

$$\text{Kategorien}(a) = \frac{K_{(W=K)}(a) + K_{(K=K)}(a) + K_{(S=K)}(a)}{K(a)}$$

Gesamtwert Der Gesamtwert der Benennungsbewertung wird im Verhältnis zur Gesamtzahl der Benennungen berechnet, um die festgelegten Grenzwerte auch bei schwankender Länge der Eingabelisten verwenden zu können (sonst führen lange Benennungslisten zu größeren Benennungsbewertungen und damit zur häufigeren Überschreitung der festgelegten Grenzwerte). $B(a)$ ist hierbei die gewichtete Bewertung aller aufgeführten Faktoren.

$$\text{Bewertung}(a) = \frac{B(a)}{n} \cdot 10$$

5.1.3 Gewichtung

Für die Gewichtung der einzelnen Faktoren aus denen sich der Gesamtwert einer Benennungsbewertung zusammensetzt, werden die gleichen Gewichtungen verwendet, die auch bei der Beziehungsbewertung (siehe Abschnitt 5.2.1) zum Einsatz kommen. Eine Ausnahme hiervon ist der in Abschnitt 4.3.2 beschriebene *Vernetzungskoeffizient*, der ausschließlich bei der Benennungsbewertung auftritt. Der Wert `qntNetworking` fließt im Standard mit einer Gewichtung von $1,5$ ein.

5.2 Beziehungsbewertung

Wichtigste Grundlage für das Strukturieren der eingegebenen Benennungsliste sind die bewerteten Beziehungen zwischen den einzelnen Benennungen. Dabei wird auf die in den vorhergehenden Kapiteln vorgestellten Methoden zurückgegriffen. Hervorzuheben ist hierbei, dass zunächst für jede Beziehungsart (Wortstämme, Kategorien, etc.) die Beziehungsstärke einzeln berechnet wird. Erst in einem späteren Schritt werden die verschiedenen Beziehungsebenen mit den unten aufgeführten Gewichtungen miteinander verrechnet.

Neben absoluten Werten für die Beschreibung einer Beziehungsstärke (Sørensen-Dice, Levenshtein) können andere Methoden unterschiedlich starke Beziehungen entdecken (z.B. gemeinsames Grundwort *vs.* gemeinsamer Wortstamm). Um diese Werte in einem späteren Schritt miteinander verrechnen zu können müssen zum einen Umformungen vorgenommen werden und zum anderen Grund- bzw. Maximalwerte festgesetzt werden.

5.2.1 Faktoren

Die Wahl der Standards für Basiswerte und Gewichtungen basiert auf Erfahrungswerten, die zum Einen mit der Qualität der Datenquellen (Synoynm-Datenbank, Wikipedia) zu tun hat, zum Anderen aber auch mit den getesteten Benennungslisten (alle aus dem Bereich *Automobile*).

Folgende Faktoren werden bei der Beziehungsbewertung berücksichtigt:

Statistische Verfahren

Sørensen-Dice-Koeffizient Der berechnete Koeffizient wird mit der Basis `cfgSørensenDiceBase` (Standard: 100) multipliziert und mit der Gewichtung `qntDiceCoefficent` (Standard: 1) multipliziert.

Levenshtein-Distanz Die Basis `cfgLevenshteinBase` (Standard: 100) wird durch die berechnete Distanz dividiert und mit der Gewichtung `qntLevenshtein` (Standard: $0,5$) multipliziert.

Wortstämme

Alle gefundenen Verbindungen werden miteinander addiert und mit der Gewichtung `qntWordStems` (Standard: $3,5$) multipliziert.

Gemeinsame Wortstämme Die Anzahl gemeinsamer Wortstämme wird durch die Anzahl an Wortstämmen des Ausgangsworts dividiert und mit der Basis `cfgPtCommonStem` (Standard: 70) multipliziert.

Gemeinsames Grundwort Bei einem gemeinsamen Grundwort wird die Beziehung mit der Basis `cfgPtCommonBaseStem` (Standard: 80) bewertet.

Wort ist Grundwort Ist ein Wort das Grundwort des anderes wird die Beziehung mit der Basis `cfgPtWordIsBaseStem` (Standard: 120) bewertet.

Kategorien

Alle gefundenen Verbindungen werden miteinander addiert und mit der Gewichtung `qntCategorial` (Standard: 3) multipliziert.

Gemeinsame Kategorie Die Anzahl gemeinsamer Kategorien wird mit der Basis `cfgPtCommonCategory` (Standard: 40) multipliziert.

Wort ist Kategorie Ist ein Wort die Kategorie des anderen wird die Beziehung mit der Basis `cfgPtWordIsCategory` (Standard: 130) bewertet.

Synonym ist Kategorie Ist das Synonym eines Worts die Kategorie des anderen wird die Beziehung mit der Basis `cfgPtSynonymIsCategory` (Standard: 60) bewertet.

Synonyme

Alle gefundenen Verbindungen werden miteinander addiert und mit der Gewichtung `qntSynonymal` (Standard: 2) multipliziert.

Gemeinsames Synonym Die Anzahl gemeinsamer Synonyme wird mit der Basis `cfgPtCommonSynonym` (Standard: 50) multipliziert.

Wort ist Synonym Ist ein Wort das Synonym des anderen wird die Beziehung mit der Basis `cfgPtWordIsSynonym` (Standard: 100) bewertet.

5.2.2 Parameter

Die Anwendung kann mit folgenden Parametern weiter eingestellt werden:

Länge der N-Gramme Über den Parameter `cfgLengthOfNgrams` (Standard: 3) kann die Länge der N-Gramme für die Berechnung des Sørensen-Dice-Koeffizienten gesteuert werden. In der Regel empfiehlt sich die Verwendung von Trigrammen ($n = 3$), in Sonderfällen kann der Wert aber auch um 1 Punkt nach oben oder unten angepasst werden.

Mindestlänge der Wortstammkandidaten Über den Parameter `cfgMinLengthOfStem` (Standard: 4) kann die Mindestlänge für Wortstammkandidaten bei der algorithmischen Wortstammsuche gesteuert werden (siehe Abschnitt 3.2.5). Erst ab dieser Länge werden Zeichenkette bei der Suche berücksichtigt. Kleinere Werte können zu falsch positiven Wortstämmen führen. Wird der Wert erhöht verringert sich sowohl die Fehler- als auch die Gesamttrefferzahl.

5.2.3 Interne Visualisierung

Um die Beziehungsstärken während der Entwicklungsphase kontrollieren zu können, wurde in die Anwendung eine Visualisierungsfunktion integriert, die alle Benennungen in einer Matrix gegenüberstellt und die Stärke eine Beziehung farblich[1] markiert (siehe Abbildung 5.1).

5.3 Gruppierung

5.3.1 Vorgehen

Um basierend auf den Beziehungsbewertungen eine strukturierte Liste zu generieren, werden zunächst Hauptgruppen gebildet. Diese Hauptgruppen basieren auf einer Rangliste der Benennungen mit den höchsten Einzelbewertungen. Alle Benennungen deren Bewertung einen bestimmten Grenzwert übersteigen bilden daraufhin eine Gruppe, die entsprechend benannt wird (Ebene 1).

Anschließend werden die am höchsten bewerteten Beziehungen, die von den Gruppen-Benennungen ausgehen und über einem bestimmten Grenzwert liegen als Gruppenmitglieder zugelassen (Ebene 2). Innerhalb dieser Gruppenmitglieder können einzelne

[1] von den Werten $0 - 100$ mit dunkler werdendem Blau, darüber mit gleich bleibendem Blau.

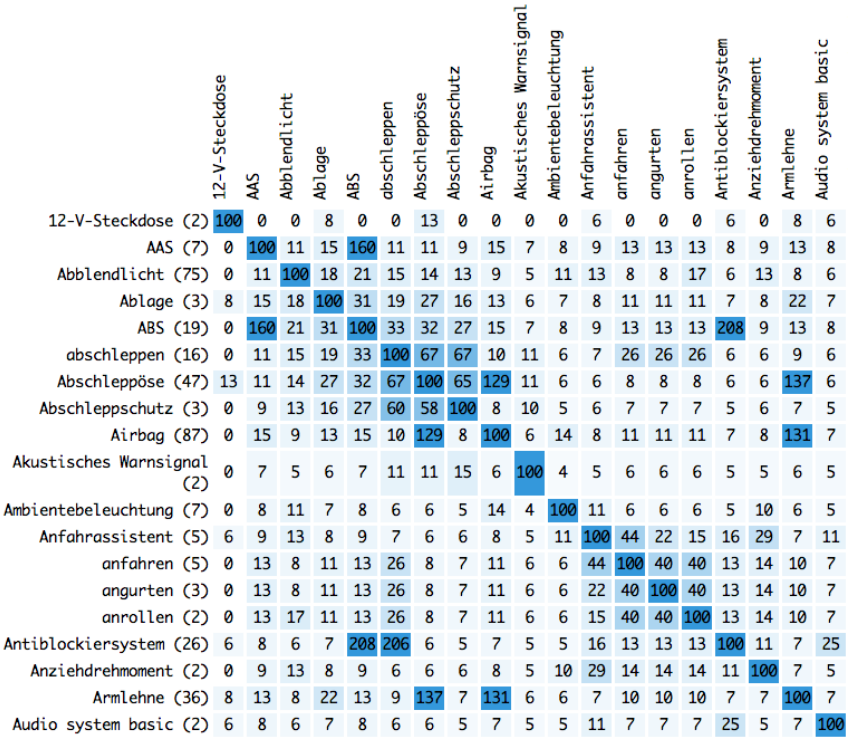

Abbildung 5.1: Ausschnitt der Matrix-Visualisierung

Benennungen als Synonym (*S*) oder über ein Kategorieverhältnis verwandt (*K*) gekennzeichnet werden (siehe Abbildung 5.2).

Ob eine Benennung gekennzeichnet wird hängt ebenfalls davon ab, ob sie einen bestimmten Grenzwert in der jeweiligen Beziehungsebene übersteigt. Nicht gekennzeichnete Benennungen können aus mehrere Gründen in einer Gruppe sein (stat. Wortähnlichkeit, verwandte Wortstämme, etc.)

5.3.2 Grenzwerte

Die Wahl der in Abschnitt 5.3.1 beschriebenen Grenzwerte basiert auf Erfahrungswerten. Generell bedeuten niedrigere Grenzwerte mehr Ergebnisse, die jedoch auch ungenauer werden. Die Grenzwerte werden im folgenden beschrieben:

Abbildung 5.2: Ausschnitt des Gruppierungsresultats

Gruppierung

Wort wird Gruppe Um eine neue Gruppe zu bilden, muss die Einzelbewertung einer Benennung den Wert `trsGroupWords` (Standard: 100) übersteigen.

Wort wird Gruppenmitglied Um Mitglied der Gruppe eines Wortes W_1 zu werden, muss die Bewertung der Beziehung von W_1 zu diesem Wort den Wert `trsSubGroupMembers` (Standard: 50) übersteigen.

Anmerkung: Dieser Wert sollte je nach Qualität der Gruppenmitglieder angepasst werden und ist stark abhängig von der übergebenen Benennungsliste. Ist Benennungsliste sehr heterogen und die Benennungen hab en wenig Gemeinsamkeiten (sowohl sprachlich als auch semantisch), so muss der Wert nach unten korrigiert werden. Das Gegenteil trifft zu, wenn sich die Benennungen sehr ähneln.

Kennzeichnung

Synonym Um innerhalb der Gruppe eines Wortes W_1 als Synonym von W_1 gekennzeichnet zu werden, muss die Synonym-Beziehung von W_1 zu diesem Wort (`relSynons`) den Wert `trsMarkerSynonym` (Standard: 100) übersteigen.

Kategorie Um innerhalb der Gruppe eines Wortes W_1 als Kategorieverhältnis von W_1 gekennzeichnet zu werden, muss die Kategorie-Beziehung von W_1 zu diesem Wort (`relCategs`) den Wert `trsMarkerCategory` (Standard: 70) übersteigen. Dabei ist nicht festgelegt, ob es sich um einen Ober- oder Unterbegriff handeln muss, es wird lediglich die Beziehungsart (*Abstraktionsbeziehung*) beschrieben.

6 Technisches Konzept

6.1 Systemaufbau

Das System wird als clientseitige Web-Applikation entwickelt, was bedeutet, dass alle Rechenoperationen im Browser des Benutzers ausgeführt werden. Eine Serverkommunikation ist nur für das Einlesen von Datenquellen notwendig (die sowohl lokal als auch auf anderen Servern liegen können). Für den Zugriff auf Webquellen (z.B. die Wikipedia-API) muss eine Internetverbindung vorhanden sein. Durch diesen Aufbau lässt sich die Applikation in jedem modernen Browser ausführen.

Als Programmiersprache kommt JavaScript zum Einsatz, als Datenaustauschformat JSON (JavaScript Object Notation). Die kommentierte Codebasis umfasst ca. 830 Zeilen objektorientierten Code. Die Eingabe des Nutzers

6.2 Datenquellen

Die Ursprünge der verwendeten Datenquellen werden in den einzelnen Kapitel ausführlich behandelt. An dieser Stelle sind nur die technischen Rahmendaten zur Übersicht festgehalten (*Einträge* bezieht sich hier auf die verwendbaren und bereinigten Suchschlüssel; ein Eintrag kann mehrere oder keine Werte zurückliefern.):

Funktion	Dateiname	Dateigröße	Einträge	Quelle
Lemmatisierung	baseforms.json	10,3 *MB*	361.267	Naber
erw. Lemmatisierung	baseforms.ext.json	12,2 *MB*	428.547	Naber
Kompositazerlegung	compounds.json	7,1 *MB*	177.843	Lemberg
Synonymfindung	thesaurus.json	15,8 *MB*	93.604	OpenThes.
Kategoriefindung	(API)	8.3 *GB*	1.732.222	Wikimedia

Lokale Datenquellen (`*.json`) werden beim Start der Anwendung in den Arbeitsspeicher des ausführenden Rechners geladen. Die Wikipedia-API wird bei Start der Strukturierung parallel zu den anderen Verfahren für jede Benennung angesteuert, die zurückgelieferten Ergebnisse werden wiederum in den Arbeitsspeicher des Rechners geladen und stehen dort zur Verarbeitung zur Verfügung.

6.3 Oberfläche

Die Oberfläche ist auf das Nötigste reduziert: sie bietet ein Textfeld zum Einfügen der Benennungsliste, eine Statusanzeige, Bedienelemente und eine Ergebnisanzeige. Bedienelemente erscheinen nur dann, wenn man sie auch benutzen kann. Die Statusanzeige zeigt den aktuellen Fortschritt in der Verarbeitung. Zusätzlich zur strukturierten Ergebnisliste (Button *Strukturieren*) kann sich der Benutzer auch die interne Matrix-Visualisierung anzeigen lassen (Button *Visualisieren*).

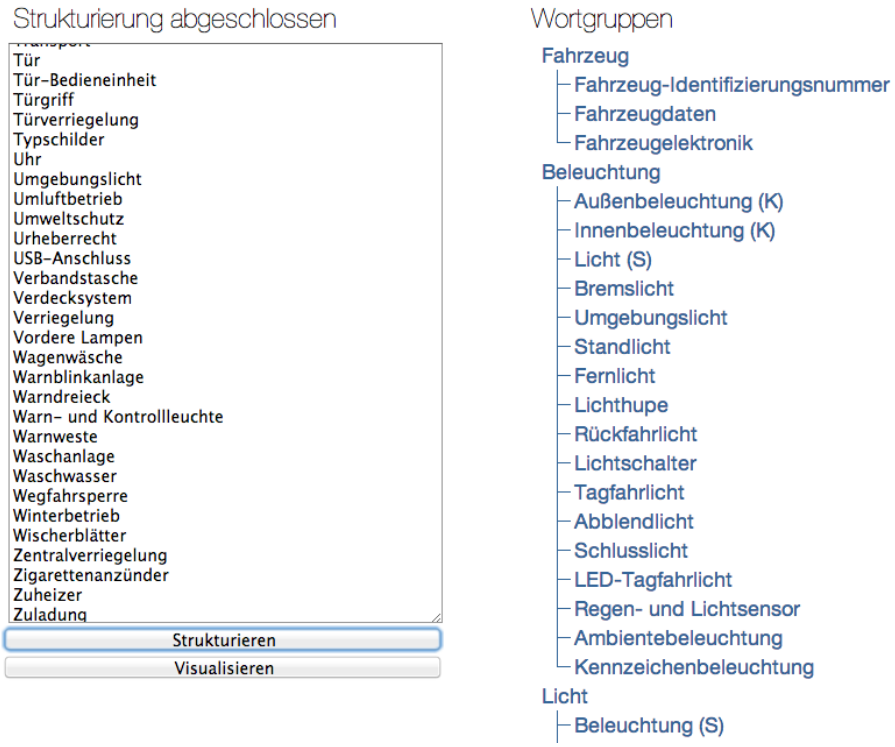

Abbildung 6.1: Screenshot der Oberfläche (Ausschnitt)

Beim Starten der Anwendung werden zunächst die lokalen Datenquellen geladen. Der aktuellen Fortschritt des Startvorgangs wird dem Nutzer in der Statusanzeige angezeigt.

Anmerkung: Bei Tests konnte beobachtet werden, dass es unter Umständen zu Problemen mit Listen kommen kann, die deutlich über 1000 Einträge lang sind. Diese Probleme sind auf das Rendering der gruppierten Liste im Browser zurückzuführen, der eigentliche Strukturierungsmechanismus kann auch mit deutlich größeren Listen umgehen (begrenzt durch den Arbeitsspeicher des Rechners).

6.4 Punktesystem

Die Anwendung basiert auf einem komplexen Punktesystem zur Bewertung von Benennungen und deren Beziehungen zueinander. Bewertet wird hier keine konkrete Größe, sondern eher eine Art Relevanz der Benennung innerhalb der übergebenen Benennungsliste. Die genaue Funktionsweise des Bewertungssystems wird in späteren Kapiteln erläutert. Alle Bestandteile des Bewertungssystems sind parametrisiert und können je nach Anforderungen an die spezifischen Inhalte angepasst werden. Eine kurze Übersicht über die Parameter findet sich hier:

```
/* config */
cfgLengthOfNgrams = 3,          // character length (n-value) of n-grams; default 3
cfgMinLengthOfStem = 4,         // minimum character length for searching word stems in compunds
cfgSørensenDiceBase = 100,      // base value for sørensen-dice factoring (default: 100)
cfgLevenshteinBase = 100,       // base value for levenshtein factoring (default: 100)
cfgPtCommonStem = 70,           // common stem in compounds (relative to total number of stems)
cfgPtCommonBaseStem = 80,       // common base stem in compounds (last stem)
cfgPtWordIsBaseStem = 120,      // word that is a base stem in compounds (last stem)
cfgPtCommonCategory = 40,       // common encyclopaedic category of two words
cfgPtWordIsCategory = 130,      // word that is a encyclopaedic category to another word
cfgPtSynonymIsCategory = 60,    // synonym of a word that is a encyclopaedic category to another word
cfgPtCommonSynonym = 50,        // common synonym of two words
cfgPtWordIsSynonym = 100,       // word that is the synonym to another word
/* quantifiers */
qntLevenshtein = 0.5,           // quantifier for dice coefficient
qntDiceCoefficent = 1,          // quantifier for dice coefficient
qntWordStems = 3.5,             // quantifier word stem relations
qntCategorial = 3,              // quantifier categorial relations
qntSynonymal = 2,               // quantifier synonymal relations
qntNetworking = 2,              // quantifier network value (word)
/* tresholds */
trsGroupWords = 100,            // treshold for word to become group heading
trsSubGroupMembers = 50,        // treshold for word to become group member
trsMarkerSynonym = 100,         // treshold for word to be markes as synonym with (S)
trsMarkerCategory = 70,         // treshold for word to be markes as synonym with (C)
```

Abbildung 6.2: Codeausschnitt Parametrisierung

7 Fazit und Ausblick

7.1 Zusammenfassung

In Laufe der Arbeit wurden verschiedene Methoden entwickelt, um Benennungen zu lemmatisieren und Beziehungen zwischen Benennungen zu finden und zu bewerten. Neben statistischen Methoden wurde die Zerlegung von Komposita in Wortstämme und deren Beziehungen sowie semantische Relationen zwischen Benennungen untersucht. Neben programmatischen Verfahren konnte bei der Umsetzung das Wissen externer Datenquellen integriert und verarbeitet werden. Darauf aufbauend konnte ein Verfahren zur Strukturierung und Kennzeichnung der Benennungen umgesetzt werden.

7.2 Fazit

Die selbst gesteckten Ziele der Arbeit konnten erreicht werden: eine prototypische Anwendung, die mit Hilfe verschiedener Methoden eine beliebige Benennungsliste strukturieren kann. Neben der eigentlichen Anwendung konnten sowohl viele »Best Practices« der Computerlinguistik erlernt, aber auch eigene Verfahren entwickelt werden (*s-Flexion*, *algorithmische Wortstammsuche*).

Das Ergebnis der Strukturierung erzielt mit wenigen Ausnahmen gute Ergebnisse und lässt noch Spielraum für ein Feintuning der Parameter. Im Praxiseinsatz muss sich zeigen, ob eine solche Anpassung für verschiedene Listen sinnvoll ist, oder ob es ein allgemeingültiges Paramterset geben kann, das unabhängig der übergebenen Benennungen zuverlässig strukturiert.

Geschwindigkeit und Ladezeiten sind gut bis befriedigend, der Performance-Flaschenhals liegt hierbei allerdings auf der Serverseite (Wikipedia). Eine Strukturierun ohne den Zugriff auf die Kategorisierung wäre wesentlich schneller, allerdings auch wesentlich ungenauer.

Durch die Entwicklung des Bewertungskonzepts und der Implementierung der Anwendung konnte eine fundierte Grundlage für weitere Entwicklungen in dieser Richtung gelegt werden.

Das Ergebnis versteht sich als Hilfswerkzeug, das Terminologen die eigentliche Arbeit der begrifflichen Ordnung zwar nicht abnehmen, aber durch eine Vorstrukturierung zumindest erleichtern kann. Die Anwendung kann einen ersten Überblick über lange Listen liefern und den Einstieg in die »Handarbeit« angenehmer gestalten.

7.3 Ausblick

Bei den verwendeten Methoden zur Beziehungsbildung können besonders im Bereich der Wortstammassoziationen noch weitere Feinheiten der deutschen Sprache berücksichtigt werden. So wäre es möglich neben dem Erkennen der Kompositumsart auch detailliertere Beziehungsarten zwischen den Wortstämmen aufzubauen. Auch die Beziehung zwischen vermuteten Grundwörten von Komposita und Kategorien anderer Wörter kann noch auf brauchbare Ergebnisse hin untersucht werden. Im Bereich der Synonymerkennung könnte eine technischere Datenquelle die Qualität der Ergebnisse erhöhen.

Im weiteren Verlauf der Entwicklung können auch die selbst entwickelten Vorgehen zum Entfernen der s-Flexion und der algorithmischen Wortstammsuche weiter verbessert werden. Dort ist es nötig, sprachliche Sonderfälle zu berücksichtigen und die Anzahl an falsch positiven Treffern auf ein Minimum zu reduzieren.

Die Oberfläche der Anwendung kann benutzerfreundlicher gestaltet werden, insbesondere was die Darstellung der Ergebnisliste angeht. Hier muss untersucht werden, welche Visualisierungen sich am besten eignet um die Strukturierung der Benennungen darzustellen. Eine Bereinigung der Ergebnisse um Dubletten und Synonyme auf erster Ebene zu vermeiden ist in Planung.

Literaturverzeichnis

Busse, Dietrich (2009): Semantik. Paderborn : Fink, LIBAC

Carstensen, Kai-Uwe (2010): Computerlinguistik und Sprachtechnologie: eine Einführung. Heidelberg : Spektrum

dante e.v. (2014): Textkorpora und Wortlisten - Übersicht und Nutzungsrechte. <http://projekte.dante.de/Trennmuster/Korpora>
[Stand: Januar 2014. Letzter Zugriff: 2014-10-06]

Donalies, Elke (2005): Die Wortbildung des Deutschen: ein Überblick. Tübingen : Narr

Drewer, Petra / Ziegler, Wolfgang (2011): Technische Dokumentation. Würzburg : Vogel

Eisenberg, Peter (Hrsg.) (2007): Duden - Richtiges und gutes Deutsch - Wörterbuch der sprachlichen Zweifelsfälle. Mannheim : Dudenverlag, Der Duden in 12 Bänden 9

Fleischer, Wolfgang / Barz, Irmhild (2007): Wortbildung der deutschen Gegenwartssprache. 3., unveränderte Auflage. Tübingen : Niemeyer

Habermann, Mechthild / Diewald, Gabriele / Thurmair, Maria (2009): Duden - Grundwissen Grammatik. Mannheim : Dudenverlag, Fit für das Bachelorstudium

Hausser, Roland (2002): Grundlagen der Computerlinguistik: Mensch-Maschine-Kommunikation in natürlicher Sprache (Vorlesungsfolien).

Kluckhohn, Kim (2004): Kleines Glossar zur Einführung in die Sprachwissenschaft (Universität Leipzig). <http://www.uni-leipzig.de/~kluck/a1/glossar.htm>
[Stand: Juli 2004. Letzter Zugriff: 2014-09-07]

Kodydek, Gabriele (2001): Möglichkeiten zur Integration der sicheren sinnentsprechenden Silbentrennung in TeX. Folien zum Vortrag auf der Tagung DANTE 2001 in Rosenheim.

Köster, Torsten Bøgh (2013): Stemming german like a pro. <https://developer.s24.com/blog/08-13-2013/german_stemming_like_a_pro.html>
[Stand: August 2013. Letzter Zugriff: 2014-09-06]

Lemberg, Werner (2014): A database of German words with hyphenation information. <http://repo.or.cz/w/wortliste.git>
[Stand: Juni 2014. Letzter Zugriff: 2014-10-06]

Lezius, Wolfgang (2000): „Morphy - German morphology, part-of-speech tagging and applications.“ In: Proceedings of the 9th EURALEX International Congress., 619–623

Naber, Daniel (2013): Deutsches Morphologie-Lexikon (Lemmatisierungs-Datei). <http://www.danielnaber.de/morphologie/>
[Stand: Dezember 2013. Letzter Zugriff: 2014-09-06]

OpenThesaurus: Über OpenThesaurus / Lizenz. <http://www.openthesaurus.de/about/index>
[Stand: k.A. Letzter Zugriff: 2014-06-28]

Perera, Praharshana / Witte, René (2005): „A Self-learning Context-aware Lemmatizer for German.“ In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. Stroudsburg, PA : Association for Computational Linguistics, 636–643

Rat für deutsche Rechtschreibung (2006): Deutsche Rechtschreibung - Regeln und Wörtverzeichnis. München/Mannheim : Rat für deutsche Rechtschreibung

Wikimedia Foundation Inc. (2014a): Levenshtein Algorithm Implementations (aus dem Wikibook Algorithm Implementations). <http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance>
[Stand: Mai 2014. Letzter Zugriff: 2014-07-01]

Wikimedia Foundation Inc. (2014b): Wikipedia:Kategorien (Richtlinien Systematik). <http://de.wikipedia.org/wiki/Wikipedia:Kategorien>
[Stand: Juni 2014. Letzter Zugriff: 2014-06-29]