Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Flexibles Berechnen von Kookkurrenzen auf strukturierten und unstrukturierten Daten

Diplomarbeit

Leipzig, Juni 2006 vorgelegt von:

Marco Büchler
geb. am 24. Juli 1978
Studiengang Diplominformatik

Vorwort

Diese Arbeit ist in Zusammenarbeit mit der *Daimler Chrysler AG, Research & Technology* in Ulm entstanden. Im Rahmen dieser Zusammenarbeit konnte ich einige nützliche Einblicke in das Extrahieren von Daten bekommen. Für diese Erfahrungen und die gute Zusammenarbeit möchte ich mich an dieser Stelle recht herzlich bedanken.

Mein Dank gilt hierbei Prof. Gholamreza Nakhaeizadeh. Er hat mir die Möglichkeit gegeben, in die Analyse von Werkstattberichten Einblick zu nehmen. Die im Rahmen dieser Tätigkeit gewonnenen Ergebnisse haben das Verständnis und somit diese Arbeit entscheidend geprägt. Weiterhin gilt mein Dank Herrn Jürgen Franke, der mir im Dezember 2005 bei einem Vorortttermin einige Hilfestellungen gegeben hat sowie Herrn Markus Ackermann, der während der gesamten Arbeit hilfreich zur Seite stand.

Weiterhin möchte ich mich bei meinen Betreuern seitens der Universität Leipzig Prof. Gerhard Heyer und Stefan Bordag bedanken. Mein besonderer Dank gilt hierbei Stefan Bordag, der mich jederzeit mit seinem Wissen unterstützt hat. Lässt sich die Qualität einer Betreuung messen, dann vielleicht einerseits in dieser Arbeit. Andererseits lässt sie sich vielleicht auch mit rund 150 eMails und rund 70 Euro Telefonkosten quantifizieren.

Abschließend gilt noch mein Dank meiner Mutter Gabriele Büchler sowie meiner Mitstudentin Sophie Pitka, die diese Arbeit Korrektur gelesen haben. Den beiden Freunden Karsten Liebert und Katrin Geist gilt mein Dank dafür, dass sie mich permanent motivert haben.

Zusammenfassung

Das Berechnen von Kookkurrenzen auf Korpora mit mehr als zehn Millionen Sätzen gestaltet sich in Bezug auf die Geschwindigkeit schnell als problematisch. Daher ist es ein Ziel dieser Arbeit, einen möglichst effizienten Speicherungsmechanismus zu entwickeln. Dabei stehen zwei Grundprobleme im Mittelpunkt.

Einerseits müssen Kookkurrenzen möglichst effizient im Hinblick auf die Laufzeit berechnet werden. Hierbei stehen vor allem eine möglichst effiziente Speicherauslastung sowie möglichst wenig Festplattenzugriffe im Vordergrund. Andererseits sollen Kookkurrenz persistent auf der Festplatte abgelegt werden, so dass beim Wachsen des Korpus nur das neue Textmaterial verarbeitet werden muss. Anschließend werden die „neuen“ Kookkurrenzen mit den bisher bestehenden Kookkurrenzen zusammengeführt. Dies ermöglicht gerade bei größeren Korpora eine effiziente Berechnung.

Mittels einer solchen Bibliothek werden im Rahmen dieser Arbeit Kookkurrenzen in ihr Spektrum bezüglich des Abstandes zerlegt. Eine solche Spektralanalyse kombiniert die herkömmliche Kookkurrenzanalyse mit Strukturinformationen eines Satzes. Hierbei ergeben sich neue Möglichkeiten der Signifikanzberechnung.

In einem Kapitel werden Signifikanzmaße behandelt. Dabei werden einerseits signifikante und unsignifikante Kookkurrenzen grafisch dargestellt. Daraus sowie auch formal werden schließlich einige Zusammenhänge abgeleitet.Weiterhin werden Kookkurrenzen auf einem zufälligen und zipfverteilten Korpus berechnet.

Während Signifikanzmaße bisher vorrangig auf der stochastischen Unabhängigkeit beruhen, wird im Rahmen dieser Arbeit gezeigt, dass es für bestimmte Aufgabenstellungen sinnvoll ist, ein solches Maß mit einer Abstandsfunktion zu kombinieren. Auf diese Weise können relevante Teilspektren einer Kookkurrenz für die Weiterverarbeitung herausgefiltert werden.

Abschließend werden Kookkurrenzen auf strukturierten XML-Texten berechnet. Als Grundlage dafür dienen Werkstattberichte eines Automobilherstellers. Ziel dieser Analyse ist, Zusammenhänge zwischen den Beschreibungen des Kunden und relevanten Reparaturen zu extrahieren. Dieser Teil der Diplomarbeit steht nicht öffentlich zur Verfügung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Für das Berechnen von Kookkurrenzen gibt es derzeit verschiedene Ansätze. Einerseits wird unterstellt, dass relevante Kookkurrenzen mit geringem Abstand zwischen zwei Wörtern auftreten. Diesbezüglich werden Kookkurrenzen in unmittelbarer Nachbarschaft (vgl. [HQW06]) oder Kookkurrenzen mit einer Fenstergröße von bis zu fünf berechnet (vgl. [CH90], [CG95], [CGHH91], [BB04]). Im Rahmen dieser Arbeit sollen Kookkurrenzen auf Satzbasis in ihr *Spektrum* bezüglich des Abstandes zerlegt werden und dementsprechend in ihrer spektralen Zerlegung gezählt werden. Dadurch können auf Satzbasis nachträglich Kookkurrenzen über jede Fenstergröße berechnet werden.

Bei Nachbarschaftskookkurrenzen würden dementsprechend nur Kookkurrenzen mit den Abständen -1 und 1 berücksichtigt werden. Bei größeren Fenstern werden die Frequenzen aus mehreren Abständen aggregiert. Bei Korpora von zehn Millionen und mehr Sätzen ist dieser Ansatz insofern interessant, da Kookkurrenzen nur noch einmal in ihrer spektralen Zerlegung gezählt werden müssen und alle kleineren Fenster sich daraus berechnen lassen. Ein mehrfaches Durchlaufen mit unterschiedlichen Fenstergrößen ist daher nicht mehr notwendig und liefert so schneller Ergebnisse.

Da Korpora in der Regel mit der Zeit stetig wachsen, ist es eine Anforderung an eine *Kookkurrenzzählmaschine*, bisher berechnete Kookkurrenzen auf der Festplatte persistent abzulegen, um beim Hinzukommen von neuem Textmaterial lediglich die zusätzliche Textmenge in das Bestehende mit einzurechnen und nicht die Kookkurrenzen auf dem gesamten Korpus neu zu berechnen. Dies ist insbesondere bei Korpora wie dem *deweb*-Korpus des Projektes „Deutscher Wortschatz“ (vgl. [Qua98]) mit rund 100 Millionen Sätzen von Vorteil.

Diese Arbeit ist in drei Kapitel sowie zwei Anhänge gegliedert. In Kapitel 1 wird auf die benutzte Datenstruktur eingegangen. Die Wahl fiel dabei auf einen Hash. Gegenstand dieses Kapitels ist eine Evaluierung verschiedener

Hashfunktionen sowie die Kombination von perfektem und offenem Hashing. Abschließend werden die Ergebnisse in zwei Tabellen ausführlich dargestellt. Mit diesem Kapitel wird die Grundlage gelegt, um Kookkurrenzen einerseits persistent abzulegen und andererseits auch spektral zerlegte Kookkurrenzen zu zählen. Schließlich fliessen die Ergebnisse in einen automatischen Konfigurationsmechanismus ein, so dass die Kookkurrenzzählmaschine *Medusa* dem Benutzer diesbezüglich den Konfigurationsaufwand abnimmt und somit leichter bedienbar ist.

Das Kapitel 3 evaluiert eine Auswahl von Signifikanzmaßen. Dabei gibt es zwei Schwerpunkte. Einerseits werden die signifikantesten und die unsignifikantesten Kookkurrenzen der Maße grafisch dargestellt. Andererseits werden Kookkurrenzen auf einem zipfverteilten Zufallskorpus berechnet. Das Ergebnis dieser Betrachtung ist ein Vergleich des Zufallskorpus mit einem korrespondierenden, natürlich-sprachlichen Korpus. Kriterien sind hierbei die *Anzahl der Kookkurrenzen*, die signifikantesten Kookkurrenzen sowie *Kookkurrenzen, deren Kookkurrenzfrequenz größer als die kleinere der beiden Wortfrequenzen* ist.

In Kapitel 4 werden schließlich die Ergebnisse einer *spektralen Kookkurrenzanalyse* vorgestellt. Hierbei werden eingangs einige ausgewählte Spektren vorgestellt und erklärt. Im zweiten Teil dieses Kapitels werden diverse *semantische Relationen* und deren Abhängigkeit bezüglich des Abstandes betrachet. Die ausführlichen Ergebnisse dieses Kapitels sind in Anhang A für die Abstände -20 bis 20 dargestellt.

Im Anhang B werden Kookkurrenzen auf einem Werkstattkorpus eines Automobilherstellers berechnet. Dabei sollen Kookkurrenzen zwischen Beschwerden des Kunden und den konkreten Fehlerbeschreibungen des Mechanikers berechnet werden. Dieser Teil der Diplomarbeit ist nicht öffentlich.

1.1 Was sind Kookkurenzen?

Eine *Kookkurrenz* (engl: co-occurrence) ist im Sinne der Automatischen Sprachverarbeitung das gemeinsame Auftreten zweier *Tokens* innerhalb eines Textfensters.

Ein *Token* entspricht dabei einer textuellen Einheit wie beispielsweise einem Wort. Je nach dem benutzten *Tokenizer* kann ein Wort, auch dessen Grundform bzw. deren *morphologische Zerlegung* (vgl. [Bor06]) sein. Die Wahl des *Tokens* hängt dementsprechend von der gestellten Aufgabe ab.

Genau wie die Wahl des *Tokenizers* muss ein Textfenster festgelegt werden. Ein Textfenster kann einerseits ein *Satz*, ein *Absatz*, eine *Seite* oder ein *Dokument* sein. Solche *Kookkurrenzen* werden auch entprechend *Satz-*,

Absatz- bzw. *Dokument-Kookkurrenzen* genannt. Andererseits kann das Fenster auch eine feste Länge l haben. Diese Form der *Kookkurrenzen* werden als *fensterbasierte Kookkurrenzen* bezeichnet (vgl. [CH90],[CG95], [BB04] und [HQW06]). Eine spezielle Form der *fensterbasierten Kookkurrenzen* sind die *Nachbarschaftskookkurrenzen*. Bei dieser Form von *Kookkurrenzen* werden jeweils der unmittelbare *linke* und *rechte* Nachbar eines Wortes beobachtet. Im Rahmen dieser Arbeit soll der Fokus auf den *Satz-Kookkurrenzen* liegen. Andere *Kookkurrenzen*, wie *fensterbasierte Kookkurrenzen* und *Nachbarschaftskookkurrenzen*, werden nur kurz behandelt.

Neben der Unterteilung der *Kookkurrenzen* nach der Wahl des *Fensters*, in welchem sie beobachtet werden sollen, können *Kookkurrenzen* auch nach der Stellung der beobachteten Wörter innerhalb der Fenster kategorisiert werden. So können beispielsweise Wörter beobachtet werden, die ausschließlich rechts von einem Wort auftreten. Diese Kookkurrenzen werden *rechte Nachbarschaftskookkurrenzen* genannt. Dementsprechend können auch *linke Nachbarschaftskookkurrenzen* beobachtet werden bzw. auch beide zusammen. Dies kann z.B. zur automatischen Wortklassenclusterung verwendet werden (vgl. [Bie06b]).

Aus dem Beispielsatz:

LERNEN AN EINEM BEISPIELSATZ.

können somit für „LERNEN“ *(LERNEN,AN)*, *(LERNEN,EINEM)*, *(LERNEN,BEISPIELSATZ)* als *rechte Nachbarschaftskookkurrenzen auf Satzbasis* extrahiert werden. Ein solches Tupel wird im Rahmen dieser Arbeit als *Kookkurrenz*-Token bezeichnet. Für das Wort „EINEM“ kann aus dem obigen Beispiel nur noch die *rechte Nachbarschaftskookkurrenz (EINEM,BEISPIELSATZ)* aus dem Satz extrahiert werden. Eine *linke Nachbarschatfskookkurrenz auf Satzbasis* für „BEISPIELSATZ“ ist beispielsweise die *Kokkurrenz (BEISPIELSATZ,LERNEN)*[1].

1.2 Motivation für Kookkurrenzen

In 1.1 wurde erklärt, was unter einer *Kookkurrenz* verstanden wird. Beim Messen von Kookkurrenzen wird das gemeinsame Auftreten zweier Wörter über einen größeren Korpus gezählt. Nachdem eine *Kookkurrenz* als *signifikant* (siehe Kapitel 3) bestimmt worden ist, kann möglicherweise daraus Wissen abgeleitet werden. Die Frage ist jedoch, warum ist dies möglich?

[1] Dabei entspricht das erste Wort dem zu untersuchenden Wort.

Neben den anerkannten Motivationen von Harris (vgl. [Har51]) und Firth (vgl. [Fir57]) sei auf eine Motivation von Rapp (vgl. [WR93], [Rap02]) hingewiesen. Demnach gehen die ersten Ansätze einer Antwort auf diese Frage bereits auf Aristoteles' *Gesetze der Assoziation* zurück (vgl. [WR93], [Rap02]). Aristoteles spricht dabei von drei verschiedene Assoziationen: der *Kontiguität*, der *Ähnlichkeit* und dem *Kontrast*. Letzteres beschreibt eine große Unähnlichkeit zweier Objekte. Das für diese Zwecke relevante Gesetz ist das Gesetz des Lernens durch *Kontiguität* (lat. contiguus: angrenzend, berührend) (vgl. [Zim92]). Dabei wird von einer bereits bestehenden Wissensbasis ausgegangen. Wird ein Objekt dieses „Wissens" mit einem neuen Objekt in *räumlicher* und *zeitlicher* Nähe hinreichend oft in Verbindung gebracht, dann wird eine Assoziation zwischen beiden Objekten in der Wissensbasis des Individuums hergestellt. Der bekannteste Versuch hierzu ist die *Konditionierung nach Pawlov*. Er hatte dabei einen Hund beim Füttern einem Klingelgeräusch ausgesetzt. Nach einiger Zeit kam der Hund bereits beim Klingelgeräusch zu seinem Freßplatz und mußte nicht mehr gerufen werden (siehe [Zim92]).

Für die Automatische Sprachverarbeitung bedeutet dies, dass Wortpaare in digitalen Texten gefunden werden sollen, die die Autoren beim Produzieren eines Korpus miteinander in Verbindung gebracht haben. Dabei wird ein implizites Lernen im Laufe des Lebens nach dem Gesetz der *Kontiguität* vorausgesetzt.

So werden die meisten Menschen im Laufe des Lebens gelernt haben, dass der Begriff *Arzt* etwas mit anderen Begriffen wie *Doktor*, *Krankenhaus* oder *Krankenschwestern* zu tun hat. Jedoch wurde in den meisten Fällen nicht gelernt, dass mit *Arzt* die Begriffe wie *Nordpol*, *Kneipe* oder *Informatiker* assoziiert werden können.

Mit dem Messen von Kookkurrenzen wird schließlich versucht, für einen bestimmten Korpus dessen signifikante Assoziationen von Begriffen zu bestimmen, die die Autoren des Korpus im Laufe ihres Lebens sammeln konnten. Es werden also die langfristig gesammelten *Ergebnisse aus Erfahrungen* eines Menschen extrahiert (vgl. [MH02]).

1.3 Semantische Relationen

Jede Sprache besteht aus einem *Vokabular* und einer *Grammatik*. Das *Vokabular* einer Sprache besteht aus den möglichen Wörtern, die benutzt werden können. Mit der alleinigen Kenntnis des *Vokabulars* könnten jedoch keine syntaktisch korrekten Sätzen zu der jeweiligen Sprache entwickelt werden.

Die *Grammatik* einer Sprache setzt die Vokabeln in einer syntaktisch korrekten Reihenfolge zusammen. Dadurch stehen bespielsweise *Adjektive* vor

Substantiven und nicht dahinter, sowie der Artikel eines Substantives vor dem Substantiv und nicht beliebig im Satz.

Aus diesen Gegebenheit lassen sich diverse relevante Relationen wie die *syntagmatische*, *paradigmatische* und *hierarchisch-paradigmatische* Relation sowie die Relation der *Derivate* definieren (vgl. [dS01], [HQW06] und [Rap02]).

Unter einer *syntagmatischen* Relation zwischen zwei Wörtern wird das signifikant häufige gemeinsame Auftreten zweier Wörter in einem Satz oder einem anderen Fenster verstanden. Der im Rahmen dieser Arbeit am häufigsten beobachtete Vertreter der *syntagmatischen* Relation ist die Beziehung zwischen einem *Adjektiv* und dementsprechenden *Substantiv* (vgl. Anhang A, Tabelle A.1).

Eine *paradigmatische* Relation existiert zwischen zwei Wörtern, wenn das Ersetzen eines Wortes in einem Satz durch ein anderes Wort den Sinn weiterhin erhält. In einem Beispielsatz

> *Am 11.06.2006 spielten die Niederlande im Leipziger* Zentralstadion *gegen Serbien-Montenegro.*

kann das Wort *Zentralstadion* beispielsweise durch *FIFA-WM-Stadion* ersetzt werden, ohne dass der Satz sinnentstellt wird. Diese Form der *paradigmatischen* Relation ist sehr streng gewählt und verletzt deshalb weder die grammatische, noch die semantische Integrität des Satzes. Zusätzlich wird auch die Bedeutung beibehalten. Ein Beispiel für eine weniger strenge *paradigmatische* Relation ist:

> *Während eines Taiwanurlaubs wurde ein Tourist von einer* Schlange *gebissen.*

Hierbei kann *Schlange* durch andere Tiere wie *Hund* oder *Tiger* ersetzt werden, ohne dass der Satz dadurch semantisch inkonsistent wird. Die Bedeutung des Satzes ändert sich jedoch. Diese Formen der *paradigmatischen* Relation sind *symmetrisch*.

Unter einer *hierarchisch-paradigmatischen* Relation werden *unsymmetrisch paradigmatische* Relationen wie Unter- bzw. Oberbegriffe verstanden. Die Relation der *Derivate* haben den gleichen Ursprung im Wortstamm (Lemma) und sind Ableitungen daraus. Beispielsweise sind *Lust* und *lustig* Derivate.

Ziel dieser Arbeit wird es im Kapitel 4 sein, den Strukturalismus von *Ferdinand de Saussure* mit der herkömmlichen Kookkurrenzanalyse zu vereinen. Dabei werden Kookkurrenzen berechnet, die Auskunft darüber geben, wie sich die beiden Wörter der Kookkurrenz in einem zu beobachtenden Satzfenster zueinander verhalten ([BH05]).

1.4 Zipfsches Gesetz

Um Kookurrenzen berechnen zu können, wird auf eine größere Textsammlung von mehreren Millionen Sätzen zurückgegriffen. Eine solche Textsammlung von Sätzen einer Sprache wird *Korpus* genannt.

Innerhalb eines Korpus treten verschiedene Wortformen unterschiedlich oft auf. *George K. Zipf* formulierte dazu das *Prinzip der geringsten Anstrengung* (*Principle of Least Effort*, vgl. [Zip49]). Demnach sind häufiger benutzte Wörter kürzer, da dies „weniger Anstrengung“ bedeutet.

Wird die Wortliste eines *Korpus* nach der *Frequenz* der Wörter also nach deren *absoluten Häufigkeit* im *Korpus* sortiert, dann gilt

$$f * r \approx k. \tag{1.1}$$

f steht dabei für die Frequenz eines Wortes im Korpus und r entspricht dem Rang des Wortes in der frequenzsortierten Wortliste. Das *Zipfsche Gesetz* besagt schließlich, dass $f * r$ konstant ist.

Eine andere Schreibweise des *Zipfschen Gesetzes* ist in Formel 1.2 dargestellt. Bei dieser Formel wird deutlich, dass sich die Frequenz eines Wortes *umgekehrt proportional* zum Rang verhält.

$$f \approx \frac{k}{r} \tag{1.2}$$

In Abbildung 1.1 ist das *Zipfsche Gesetz* in einem doppelt logarithmischen *Rang-Frequenz-Plot* dargestellt.

Der Plot zeigt näherungsweise eine Gerade. Werden beide Seiten aus Formel 1.2 logarithmisiert, dann kann daraus die Formel 1.3 abgeleitet werden.

$$\log f \approx \log \frac{k}{r} = \log k - \log r \tag{1.3}$$

Wobei das Minus aus Formel 1.3 dem Anstieg des Plots aus Abbildung 1.1 entspricht. Bei realen Korpora ist dieser Wert jedoch nicht 1, sondern $1 + c$, wobei c ein kleiner positiver Wert ist.

Bezogen auf Kookkurrenzen gilt das *Zipfsche Gesetz* ebenfalls. Jedoch muss die Formel 1.2 zu der Formel 1.4 abgeändert werden ([Wik05]).

$$f \approx \frac{k}{r^2} \tag{1.4}$$

Eine Auswirkung dieser Änderung von Formel 1.4 bezüglich Formel 1.2 besteht in der Anzahl der Wörter bzw. der Kookkurrenzen, die einmal vorkommen. Während nach dem *Zipfschen Gesetzes* für Wortformen etwa die

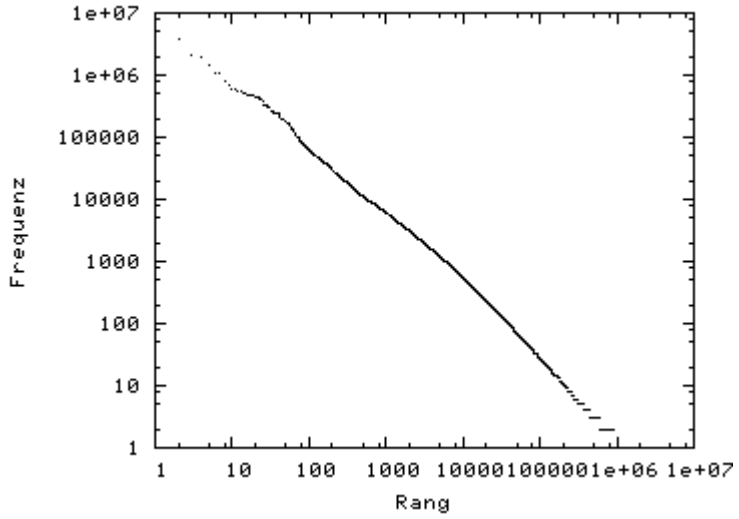

Abbildung 1.1: Rang-Frequenz-Plot einer deutschsprachige Wortliste aus 35 Millionen Sätzen, wobei beide Achsen logarithmisch skaliert sind

Hälfte aller Wortformen einmal auftauchen (vgl. [Zip49], [FS03], [FS01], [New05], [Tes92]), treten nach dem *Zipfschen Gesetz* für Kookkurrenzen etwa 75% aller Kookkurrenzen einmal auf. Diese Information ist im Kapitel 2 von besonderer Bedeutung, da dies bereits impliziert, dass es sehr viele aber dafür auch sehr viele seltene Kookkurrenzen geben wird.

Kapitel 2

Zählen von Kookkurrenzen

Die Performanz des Zählens von Kookkurrenz hängt von verschiedenen Faktoren ab. Dazu gehören Art bzw. Typ von Kookkurrenzen, sowie die Fenstergröße, in welchem Kookkurrenzen beobachtet werden sollen, nicht zuletzt aber auch die gewählte Datenstruktur. Die Fenstergröße kann einerseits dynamisch ein Satz sein. Hierbei ändert sie sich von Satz zu Satz. Andererseits kann die Fenstergröße fest gewählt werden. Diese Fenster sind in der Regel kleiner als die Fenster für die Satzkookkurrenzen.

Unter Typ oder Art von Kookkurrenzen (siehe Kapitel 1.1) werden in dieser Arbeit beispielsweise Kookkurrenzen in linker bzw. rechter Nachbarschaft oder linke und rechte Nachbarschaft zusammen verstanden. Dies ist insofern von Bedeutung, da durch je nach Art von Kookkurrenz unterschiedlich viele Kookkurrenzen aus einem bestimmten Fenster extrahiert werden können. Sei die Länge eines Satzes durch n beschrieben, dann hat das erste Wort in einem Satzfenster $n-1$ rechte Nachbarn. Das zweite Wort hat entsprechend $n-2$ rechte Nachbarn. Das $(n-1)$-te Wort hat schließlich nur noch das letzte Wort als rechten Nachbarn. Aus einem Satz der Länge n werden somit

$$
\begin{aligned}
&= n-1+n-2+n-3+...+n-(n-1) \\
&= \sum_{i=1}^{n-1} i \\
&= \frac{(n-1)(n-1+1)}{2} \\
&= \frac{(n-1)*n}{2}
\end{aligned}
$$

Kookkurrenz-Tokens (siehe Definition in Kapitel 1.1) erzeugt. Für die linken Nachbarschaftkookurrenzen (siehe Def. 1.1) gilt das Gleiche. Bei Satz-

kookkurrenzen (siehe Def. 1.1), bei denen das Vorkommen eines Wortes mit jedem anderen Wort in einem Satz gezählt werden, ergeben sich somit $n * (n-1)$ verschiedene Kookkurrenzen. Bei einer durschnittlichen Satzlänge von $n = 15$ werden dementsprechend $15 * (15 - 1) = 15 * 14 = 210$ *Kookkurrenz-Tokens* erzeugt. Beträgt die Satzlänge hingegen $n = 20$, so müssen bereits $20 * (20 - 1) = 20 * 19 = 380$ Tokens gezählt werden. Erhöht sich also die Satzlänge um 33% von 15 auf 20, so erhöhen sich bei den Satzkookkurrenzen die gefundenen Tokens um etwa 81% von 210 auf 380. Der Grund dafür liegt in dem quadratischen Verhalten der Satzkookkurrenzen bezüglich der Satzlänge.

Es kann demnach erwartet werden, dass sehr viele Kookkurrenzen aus einem Korpus erzeugt werden können. Bei einem Korpus mit 11,35 Millionen Sätzen und einer durschnittlichen Satzlänge von 32 werden somit $32 * (32 - 1) * 11.350.000 = 992 * 11,35 * 10^6 = 1,12592 * 10^{10}$ *Kookkurrenz-Tokens* erzeugt. Die rund 11 Milliarden Tokens verteilen sich auf etwa 1,5 Milliarden Kookkurrenzen, für die eine möglichst performante Datenstruktur gefunden werden muß. In diesem Kapitel werden nachfolgend verschiedene Datenstrukturen kurz vorgestellt und es wird erklärt, warum im Rahmen von *Medusa*[1] die Wahl auf einen Hash fiel.

Ziel diese Kapitels wird es sein, ein *Hashsystem* zu entwickeln, das aus einem *perfekten* und *offenen* Hash besteht. Dadurch soll die Anzahl der Kollisionen reduziert werden. Um die riesige Anzahl von Kookkurrenzen bewältigen zu können, müssen einige Optimierungen gemacht werden. Relevante Kriterien dafür sind

- *CPU-Zeit*,
- *RAM-Auslastung* und
- *Sekundärspeicherverbrauch*.

Die *CPU-Zeit* kann durch einen schnellen Algorithmus (vgl. Kapitel 2.3.2) und durch einen modifizierten *Modulo*-Operator (vgl. 2.5) reduziert werden.

Eine Optimierung der *RAM-Auslastung* kann durch eine Hashfunktion gewährleistet werden, die die Daten möglichst gleichmäßig über den Hashbereich verteilt. Ein Maß für das Streuen von Daten ist der *Avalanche-Effekt* (vgl. 2.3.1). Wenn der RAM-Hash voll ist, wird dieser auf die Festplatte ausgelagert. Durch einen guten *Avalanche-Effekt* können die Festplattenzugriffe um bis zu 20% reduziert werden.

[1] *Medusa* ist eine im Rahmen dieser Diplomarbeit entwickelte Bibliothek, die unter anderem Kookkurrenzen zählen kann.

Ferner soll die Möglichkeit bestehen, dass das Hashformat so gewählt wird, dass neben Kookurrenzen zwischen zwei Wörtern auch Kookkurrenzen zwischen mehreren Wörtern (vgl. [QW02]), spektrale Zerlegungen von Kookkurrenzen (vgl. Kapitel 4) sowie zwischen N-Grammen (vgl. [GB]) berechnet werden können.

2.1 Datenstrukturen

Um Kookkurrenzen messen zu können, wird eine Datenstruktur benötigt, die *Schüssel-Wert*-Paare speichern kann. In den *Schüssel* müssen dabei die Wortnummern der beiden Wörter einfließen. Der *Wert* entspricht der Frequenz einer Kookkurrenz. In Kapitel 1.4 wurde gezeigt, dass auch für Kookkurrenzen das *Zipfsche Gesetz* gilt. Dies bedeutet, dass eine Datenstruktur stark ungleichmäßig verteilte Daten speichern muss. So werden beispielsweise für ein hochfrequentes Wort wie *der* wesentlich mehr Kookkurrenzen gefunden als für ein Wort, welches nur einmal vorgekommen ist.

Eine letzte Anforderung betrifft die Anzahl der Kookkurrenzen. Am eingangs gezeigten Beispiel wurde vorgerechnet, dass bei rund 11 Millionen Sätzen etwa 1,5 Milliarden Kookkurrenzen gezählt werden. Dies kann erst ab etwa 8 GB Arbeitsspeicher komplett im RAM gerechnet werden. Da davon auszugehen ist, dass nach dem derzeitigen Stand der Technik ein Computer keine 8 GB Arbeitsspeicher besitzt bzw. bei größeren Korpora auch 8 GB RAM nicht ausreichen, muss eine Datenstruktur gewählt werden, die den Hauptspeicher möglichst gut ausnutzt. Unter der Annahme, dass ein handelsüblicher PC 1GB Arbeitsspeicher hat, soll im Folgenden davon ausgegangen werden, dass etwa 850 MB für die Datenstruktur benutzen werden können. Ein mögliches Auslagern auf die Festplatte wird zunächst vernachlässigt[2]. Unter diesen Bedingungen werden in den nächsten Absätzen verschiedene Datenstrukturen betrachtet.

Es gibt zahlreiche Datenstrukturen wie *Bäume*, *Tries*, *Listen* und *Hashes*. In Härder und Rahm (vgl. [HR01, Seite. 180]) werden diese Datenstrukturen in drei Klassen, *sequentielle*, *baumstrukturierte* und *gestreute* Speicherungsstrukturen, eingeteilt.

Zu den *sequentiellen* Strukturen zählen die *sequentielle Liste* und die *gekette Liste*. Diese Datenstrukturen sind für das fortlaufende Speichern geeignet. Da diese Datenstrukturen *O(n)* Zugriffe auf bereits gefundene Kookkurrenzen bietet, soll die Datenstruktur an dieser Stelle nur erwähnt und nicht

[2] *Medusa* lagert den Hash temporär auf der Platte aus, wenn der Hash im RAM voll ist. Nachdem alle Kookkurrenzen berechnet worden sind, werden alle temporären Dateien zu einem großen persistenten Festplattenhash zusammengeführt.

weiter betrachtet werden.

Binäre Suchbäume, *Mehrwegbäume* und *Digitalbäume* sind *baumstrukturierte* Datenstrukturen. Bäume basieren darauf, dass von einem Wurzelknoten ausgehend jeder andere Knoten erreicht werden kann. Je nach Art des Baumes kann so die durchschnittliche Anzahl der Zugriffe für einen gesuchten Knoten auf $O(\log n)$ reduziert werden, wobei n für die Anzahl der Knoten im Baum steht. Dies gilt jedoch nur für balancierte Suchbäume. Im *Worst Case* kann ein solcher Baum auch zu einer einfachen verketteten Liste entarten. Der *ConceptComposer*[3] basiert auf einem ternären Baum. Ein solcher Baum hat die Ordnung 3. Dies bedeutet, dass bis auf die Blätter jeder Knoten auf drei andere Knoten zeigt. Dabei werden die beiden äußeren Knoten aus Abbildung 2.1 wie bei einem binären Baum benutzt, um das erste der beiden Wörter einer Kookkurrenz zu finden. Abbildung 2.1 zeigt einen kleinen Baum, wie ihn der *ConceptComposer* benutzt. Dabei steht die Wurzel für das Wort, welches als erstes eingefügt worden ist. Würde eine sortierte Liste in diesen Baum einfügt werden, so entspricht der daraus resultierende Baum einer verketteten Liste. Die beiden äußeren Söhne entsprechen demnach einem binären Baum. Der mittlere Sohn eines Knotens zeigt auf eine verkettete Liste. In dieser Liste werden alle Kookkurrenzen zu dem Wort gespeichert, welches auf diese Liste zeigt.

Aus einem auf diese Weise erzeugten Baum wie in Abbildung 2.1 würden somit die Kookkurrenzen *ist-im (5)*, *ist-in (12)*, *ist-zu (3)* und *zu-ist (3)* gespeichert werden. Der Nachteil dieses Ansatzes bzw. aller *baumstrukturierten* Datenstrukturen ist die schlechte Speicherausnutzung im RAM. Der *ConceptComposer* benötigt pro Knoten fünf Integer-Variablen für die Wortnummer (in Abbildung 2.1 wurden stattdessen die Wörter selbst dargestellt), die Frequenz sowie drei Söhne. Dafür werden insgesamt 20 Byte pro Knoten benötigt. Davon sind aber nur 8 Byte (Wortnummer und Frequenz) relevante Informationen. Für die drei Söhne müssen 12 Byte/Knoten aufgebracht werden. Damit werden nur 40% des Hauptspeichers effektiv genutzt. Die restlichen 60% des Arbeitsspeichers enthalten nur Zeigerinformationen. Im oben genannten Beispiel, in welchem 850 MB Arbeitsspeicher für Datenstrukturen benutzt werden können, würde dies bedeuten, dass bei 20 Byte/Knoten und voller Auslastung der 850 MB rund 47 Millionen Kookkurrenzen zeitgleich im RAM gezählt werden können.

Die schlechte Ausnutzung des Arbeitsspeichers der *baumstrukturierten* Datenstrukturen kann durch einen Hash verbessert werden. Ein Hash benö-

[3]Der *ConceptComposer* ist ein Programm, welches unter anderem Kookkurrenzen zählen kann. Es wird derzeitig von der Abteilung für Automatische Sprachverarbeitung an der Universität Leipzig benutzt.

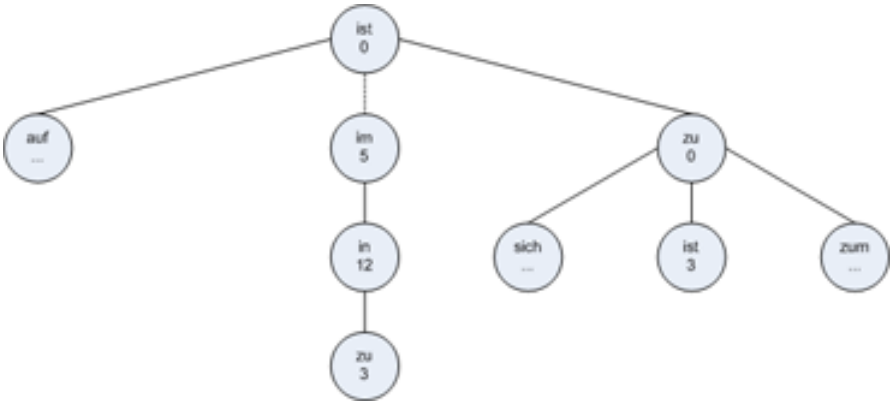

Abbildung 2.1: *baumstrukturierte* Datenstruktur zum Zählen von Kookkurrenzen

tigt keine Zeigerinformationen. In diesem Kapitel wird gezeigt, dass durch gute Hashfunktionen performant mit einer Auslastung von 90% des Hashbereichs Kookkurrenzen gezählt werden können. Auf das obige Beispiel mit 850 MB Arbeitsspeicher bezogen, bedeutet dies, dass pro Kookkurrenz 12 Byte gespeichert werden müssen. Zweimal vier Bytes werden für die beiden Wortnummern und weitere vier Bytes für die Frequenz benutzt. Somit können rund 71 Millionen Kookkurrenzen bei einer 90%igen Auslastung des Hashes im Hauptspeicher gezählt werden. Im Folgenden werden weitere Optimierungen vorgestellt, durch welche bis zu 113 Millionen Kookkurrenzen zeitgleich im RAM gezählt werden können.

2.2 Grundlagen des Hashings

Unter einem Hash kann im einfachsten Fall ein Array der Länge n verstanden werden, in welchem die Objekte gespeichert werden können. Dieses Array wird auch *Hashtabelle* genannt. Gegenüber einer Liste, in welcher die Daten, vom Index $i = 0$ beginnend, fortlaufend eingefügt werden, wird der Index i des Array mittels einer Funktion $h : \mathcal{K} \rightarrow 1, ..., n$ berechnet, die auch als *Hashfunktion* bezeichnet wird. Dabei ist $\mathcal{K} \subseteq T \times T$ das Kreuzprodukt der Menge T aller Wörter bzw. Wortnummern. Für Kookkurrenzen n-ter Ordnung ist K dementsprechend mit $\mathcal{K} \subseteq T^n$ definiert.

Im Rest dieses Kapitels werden die Ergebnisse reflektiert, die im Rahmen der Entwicklung von *Medusa* gewonnen worden sind. Im Besonderen stehen die beiden Gütekriterien *bestmögliche Auslastung* und *maximale Geschwindigkeit* im Mittelpunkt. Beide Kriterien verhalten sich konträr zueinander, so

dass das Verbessern des einen Kriteriums ein Verschlechtern des anderen zur Folge hat. Es gilt also, die bestmögliche Kombination beider Gütekriterien zu finden.

2.3 Wahl der passenden Hashfunktion

Eine Hashfunktion hat die Aufgabe, aus einem Definitionsbereich $\mathcal{K}$ möglichst gleichmäßig auf einen Wertebereich $1, 2, ..., n-1, n$ zu streuen. Wird eine Gleichverteilung der Daten aus dem Wertebereich bezüglich ihrer Frequenz zugrunde gelegt, dann genügt es, eine triviale Hashfunktion wie

$$h(k) = (k_1 * k_2 * \cdots * k_n) \bmod s \quad mit \quad (k_1, k_2, \ldots, k_n) = k \in \mathcal{K} \subseteq T^n \quad (2.1)$$

zu benutzen, die die Wortnummern der Kookkurrenz miteinander multipliziert. Da sich die Wörter eines Korpus aber nicht gleichverteilen, sondern zipfverteilt sind, ist die Benutzung einer trivalen Hashfunktion wie die *Multiplikation* in Kombination mit einer *Modulo-Operation* problematisch, da sowohl die *Multiplikation* als auch die *Modulo-Operation* zu stark von der Verteilung der Schlüssel aus dem Wertebereich abhängen (siehe Tabelle 2.2 in Kapitel 2.3.1).

2.3.1 Avalanche-Effekt

Aus Gründen der Geschwindigkeit und des Platzverbrauchs werden Wörter als Wortnummern dargestellt. In der Regel sind diese Wörter nach einem Kriterium, wie alphabetischer Ordnung oder Frequenz, sortiert. Gerade letzteres birgt jedoch einige Gefahren. Wird 1_{10} in der Darstellung zur Basis 2 betrachtet, dann wird im 32-Bit-Integer-Bereich 31mal die 0 und nur einmal die 1 gesetzt.

$$00000000000000000000000000000001_2 = 1_{10}$$

Für die Betrachtung der Zahl 3_{10} hat sich nur das vorletzte Bit geändert.

$$00000000000000000000000000000011_2 = 3_{10}$$

Würde nun eine Kookkurrenz zwischen diesen beiden Wörter gezählt werden, dann hätte eine Hashfunktion nur drei gesetzte Bits dazu zur Verfügung. Eine Multiplikation beider Zahlen würde wieder 3_{10} zurückliefern.

$$\begin{aligned} &00000000000000000000000000000001_2 \\ *&00000000000000000000000000000011_2 \\ =\ &00000000000000000000000000000011_2 \\ &= 3_{10} \end{aligned}$$

Für eine Addition beider Zahlen würden sich in diesem Beispiel wenigstens einige Bits ändern.

$$\begin{aligned} & 00000000000000000000000000000001_2 \\ *& 00000000000000000000000000000011_2 \\ = & 00000000000000000000000000000100_2 \\ & = 4_{10} \end{aligned}$$

Da bei einer frequenzsortierten Wortnummernvergabe gerade die häufigsten Wörter durch sehr kleine Zahlen und damit durch relativ wenig gesetzte Bits repräsentiert werden, muss also an eine Hashfunktion die Anforderung gestellt werden, dass sie auch 1-Bits „produzieren“ kann. Die Fähigkeit, auf Änderung eines Bits in der Eingabe, Bits in der Ausgabe zu ändern, wird *Avalanche-Effekt* genannt.

Von einem guten *Avalanche-Effekt* wird gesprochen, wenn sich in der Eingabe ein Bit ändert und aufgrund dessen die Hashfunktion 50% der Ausgabebits modifiziert. Daher wird der *Avalanche-Effekt* auch *Lawineneffekt* genannt. Der *Avalanche-Effekt* mißt demnach unter anderem wie gut eine Hashfunktion Bits setzen kann, wenn in der Eingabe nur wenige Bits gesetzt gewesen sind (vgl. [Wik05]).

Im Rahmen dieser Arbeit wird von einem *optimalem Avalanche-Effekt* gesprochen, wenn sich 50% der Bits in der Ausgabe ändern. Dies läßt sich auch mittels der Entropie formal belegen. Die Entropie ist ein Maß für die Unsicherheit eines Ereignisses. Tritt dieses Ereignis schließlich ein, dann ist der entsprechende „Informationsgewinn“ umso größer (vgl. [Top73]).

Die zu benutzende Entropiefunktion $H(p, 1-p)$ besteht aus zwei Eingangswahrscheinlichkeiten. p steht für die Wahrscheinlichkeit, dass ein Bitwechsel stattfindet. $H(p, 1-p)$ ist somit definiert als

$$H(p, 1-p) = -p * \log(p) - (1-p) * \log(1-p). \tag{2.2}$$

Das Maximum der Entropiefunktion $H(p, 1-p)$ liegt, wie in Abbildung 2.2 dargestellt, bei einer Wahrscheinlichkeit von 50% (vgl. [Sha48], [Top73], [Völ91] und [Mac03]).

Aus Sicht des Hashings bedeutet eine Wahrscheinlichkeit des Bitwechsels von 50%, dass die Entropie bzw. die „Verunsicherung“ des Ausgangs am größten ist. Dies bedeutet ferner, dass die Fähigkeit zu streuen, ihr Optimum erreicht.

Um den *Avalanche-Effekt* zu messen, wurden die in den Tabellen 2.1 und 2.2 möglichen Hashfunktionen auf diesen Effekt untersucht. Dazu wurden die Hashwerte zweier Eingaben, die sich in einem Bit unterscheiden, miteinander verglichen. In Tabelle 2.1 sind die Ergebnisse der ersten 20 Millionen

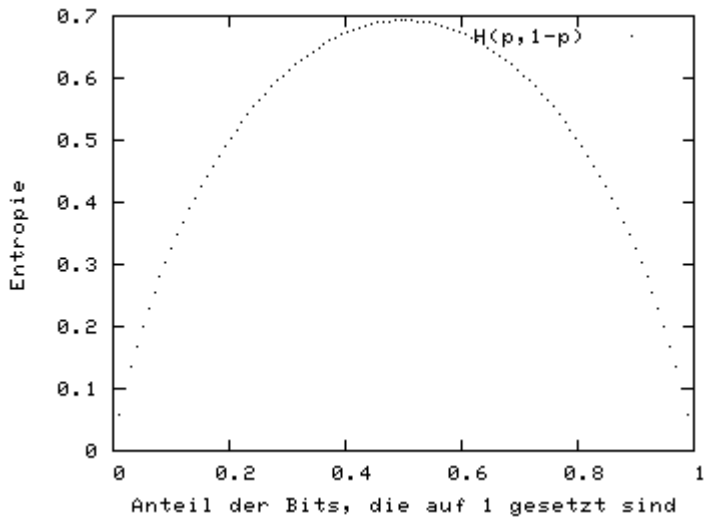

Abbildung 2.2: Entropie H(p,1-p) als Maß für die Unsicherheit des Ausgangs eines Ereignisses (*p* entspricht der Wahrscheinlichkeit, dass ein Bitwechsel stattfindet. *1-p* stellt die Wahrscheinlichkeit dafür dar, dass kein Bitwechsel stattfindet.)

Wortnummern für diesen Effekt dargestellt. Für die Manipulation eines Bits wurde das letzte Bit der Eingabe ausgesucht, welches darüber entscheidet, ob die Eingabezahl gerade oder ungerade ist. Dementsprechend wurde der Hashwert einer geraden Zahl berechnet. Anschließend wurde das letzte Bit der Eingabezahl gesetzt. Zum Vergleich hat sich damit in der Eingabe nur ein Bit verändert. Von dieser modifizierten Eingabe wurde ebenfalls der Hashwert berechnet. Die beiden, auf diese Weise erhaltenen Hashwerte, wurden anschließend *XOR*-verknüpft. Im Bit i dieses Ergebnisses wurde durch die *XOR*-Veknüpfung eine 1 gesetzt, wenn ein Bitwechsel an der Position i stattgefunden hat. Ansonsten wird eine 0 gesetzt. Anschließend wurden die letzten 8 Byte[4] des Hashwertes byteweise auf gesetzte Bits untersucht. Bei einem guten *Avalanche-Effekt* wurden so im Schnitt 4 von 8 Byte gezählt.

Tabelle 2.1 zeigt den, auf diese Weise berechneten, *Avalanche-Effekt* für die letzten 8 Byte. Dabei steht die Spalte *1* für das letzte Byte, Spalte *2* für das vorletzte Byte usw. Eine reine *XOR*-Verknüpfung würde lediglich eine Änderungen im letzten Byte nach sich ziehen (vgl. Tabelle 2.1). Ebenfalls zeigt sich, dass der *Adler32*-Algorithmus (vgl. [Wik05]) für das letz-

[4]Aus Gründen, die in Kapitel 2.5 noch erklärt werden, genügt es, die letzten 8 Byte zu betrachten.

Hashfunktion	Avalanche-Effekt für die letzten 8 Byte in %							
	1	2	3	4	5	6	7	8
XOR	12,5	0	0	0	- - -	- - -	- - -	- - -
CRC32	50,0	62,5	62,5	50	- - -	- - -	- - -	- - -
Adler32	24,9	0,5	24,6	0,3	- - -	- - -	- - -	- - -
MD2	50,0	50,0	49,9	50,0	49,9	50,0	49,9	50,0
MD5	49,9	49,9	49,9	50,0	50,0	50,0	50,0	49,9
SHA-160	50,0	49,9	50,0	50,0	50,0	49,9	50,0	49,9
SHA-256	49,9	50,0	50,0	49,9	49,9	50,0	50,0	50,0
SHA-384	49,9	49,9	49,9	50,0	49,9	49,9	50,0	49,9
SHA-512	50,0	50,0	50,0	49,9	49,9	50,0	49,9	50,0

Tabelle 2.1: gemessener *Avalanche-Effekt* für die ersten 20 Millionen Zahlen, basierend auf einer Änderung des letzten Bits in der binären Zahlendarstellung

te und drittletzte Byte einen sehr schlechten *Avalanche-Effekt* hat. Für das zweit- und viertletzte Byte ist dieser Effekt kaum noch messbar. Der *CRC32*-Algorithmus ([Wik05]) misst für das vorletzte und drittletzte Byte sehr starke Änderungseffekte.

Eine Änderung von jeweils 62,5% bedeutet jedoch nicht, dass dieser Algorithmus dadurch besser ist. Das Ziel des *Avalanche-Effektes* in der Kryptologie ist, dass das Knacken einer verschlüßelten Nachricht möglichst schwer ist. Wird ein niedriger Effekt gemessen, dann ist die verschlüsselte Nachricht leicht zu knacken. Andererseits wird ein hoher *Avalanche-Effekt* gemessen, dann bedeutet dies, dass die Nachricht nur stärker bitweise invertiert wird. Daher ist es auch das Ziel, Effekte von 50% Bitänderungen zu messen. Für das Hashen bedeutet dies, dass bei einem niedrigen *Avalanche-Effekt* ähnliche $k, k' \in \mathcal{K}$ in der Hashtabelle lokal clustern. Bei einem entsprechend hohen Effekt clustern ähnliche $k, k' \in \mathcal{K}$ nur an einer anderen Stelle. Optimale Effekte (siehe Abbildung 2.2) von 50% streuen die Daten bestmöglich über den gesamten Hashbereich. In Tabelle 2.1 ist der *Avalanche-Effekt* für die Algorithmen *SHA-160*, *SHA-256*, *SHA-384*,*SHA-512*, *MD2* und *MD5* (vgl. [Sch96] und [Bau94]) dargestellt.

In einem zweiten Test zum *Avalanche-Effekt* wurden 100 Millionen zufällige und zipfverteilte Kookkurrenzen unter Verwendung der Formel aus [New05][5] erzeugt. Der wesentliche Unterschied zum ersten Test besteht demnach darin, dass die Daten nicht mehr gleich- sondern zipfverteilt sind. Eine

[5]Siehe Seite 3 Fußnote 3.

der beiden Wortnummern wurde wieder wie im obigen Beispiel gerade gewählt, so dass mittels setzen des letzten Bits eine kontrollierte Bitänderung vorgenommen werden kann.

Hashfunktion	Avalanche-Effekt für die letzten 8 Byte in %							
	1	2	3	4	5	6	7	8
Addition	25,2	0,04	$7.8 * 10^{-6}$	0	- - -	- - -	- - -	- - -
Multiplikation	42,4	9,9	0,7	0,04	- - -	- - -	- - -	- - -
CRC32	62,5	0,0	50,0	50,0	- - -	- - -	- - -	- - -
Adler32	24,6	0,02	24,5	0,5	- - -	- - -	- - -	- - -
MD2	49,9	49,4	50,0	49,8	49,7	50,2	49,7	50,7
MD5	50,2	50,3	50,2	49,7	49,3	49,8	49,6	50,1
SHA-160	50,3	49,8	50,2	49,6	50,7	50,2	50,0	50,2
SHA-256	50,6	50,0	50,3	49,7	49,9	49,5	50,2	49,5
SHA-384	49,9	49,9	49,9	50,0	49,9	49,9	50,0	49,9
SHA-512	49,5	49,9	50,2	49,7	49,7	50,7	49,4	49,5

Tabelle 2.2: Avalanche-Effekt

Tabelle 2.2 zeigt die Ergebisse dieses Tests. Die eingangs erwähnten Hashfunktionen *Addition* und *Multiplikation* von Wortnummern erreichen sehr schlechte Werte. Weiterhin werden Bitänderungen nicht über den gesamten Hashbreich gestreut, sondern sind bei dem Byte am größten, in welchem die Bitänderung gemacht worden ist. Je größer der Abstand von diesem Byte ist, umso geringer wird ein *Avalanache-Effekt* meßbar. Bei dem *CRC32*-Algorithmus werden im vorletzten Byte sogar keine Änderungen gemessen. Die Hashfunktionen, die aus der Kryptologie kommen, zeigen im Vergleich zu der gleichverteilten Eingabe nur geringfügige Änderungen.

Aus Sicht des *Avalanche-Effektes* müsste dementsprechend die Wahl der Hashfunktion auf *MD2*, *MD5*, *SHA-160*, *SHA-256*,*SHA-384* oder *SHA-512* fallen.

2.3.2 Hashfunktionen

Im Rahmen von *Medusa* wurden Hashfunktionen evaluiert, die einerseits zum Berechnen von *Checksummen* und andererseits in der Kryptologie zum *Erstellen von Signaturen*, die für das Verschlüsseln von Daten benutzt werden. Zur ersten Gruppe von Hashfunktionen zählen beispielsweise Algorithmen wie *CRC16*, *CRC24*, *CRC32*, *CRC32c*, *CRC64* oder *Adler32*. Diese Algorithmen basieren auf der Berechnung von Polynomen. Zur zweiten Gruppe zählen

Hashfunktionen wie *MD2*, *MD5*, *SHA-160*,*SHA-256*,*SHA-384* und *SHA-512*. Diese Algorithmen basieren auf einer Zerlegung der Eingabe in verschiedene Blöcke. Danach werden diese Blöcke iterativ verarbeitet. Die genaue Funktionsweise dieser Algorithmen soll an dieser Stelle vernachlässigt werden. Es sei jedoch auf externe Literatur wie [Sch96], [Bau94] und [Wik05] verwiesen.

Ein paar der eben genannten Algorithmen wurden im Rahmen von *Medusa* genauer untersucht. Tabelle 2.3 stellt ausgewählte Funktionen mit einigen Informationen dar. Die Länge des Hashwertes ergibt sich aus dem jeweiligen Algorithmus. Die *benötigte Zeit* entspricht der Zeit, die gebraucht worden ist, um 100 Millionen Hashwerte mittels des jeweiligen Algorithmus zu berechnen. Die beiden Wörter, die in diesem Szenario „gemeinsam miteinander auftreten“ sollen, wurden mittels der Formel aus [New05] berechnet. Die letzte Spalte dieser Tabelle zeigt den Skalierungsfaktor in Bezug auf den schnellsten Algorithmus.

Hashfunktion	Länge des Hashwertes in Bytes	benötigte Zeit in s	Faktor
Adler32	4	45	1
CRC32	4	53	1,18
MD2	16	1046	23,3
MD5	16	145	3,2
SHA-160	20	213	4,8
SHA-256	32	289	6,4
SHA-384	48	1050	23,4
SHA-512	64	1062	23,6

Tabelle 2.3: ausgewählte Algorithmen, die als Hashfunktion benutzt werden können

Tabelle 2.3 zeigt, dass es aus Sicht der Performance eine klare Trennung zwischen den *Checksummen*- und den *Kryptographie*-Algorithmen gibt. Während der *Adler32*- und der *CRC32*-Algorithmus beide fast gleich schnell sind, benötigen alle *Kryptographie*-Algorithmen um eine Größenordnung mehr Zeit für das gleiche Problem.

Neben dem Gütekriterium *maximale Geschwindigkeit* wird im Rahmen dieser Arbeit auch das Kriterium *bestmögliche Auslastung* definiert. Aus Sicht der Geschwindigkeit müßte die Wahl auf den *Adler32*- oder bestenfalls noch auf den *CRC32*-Algorithmus fallen. Diese Betrachtungen berücksichtigen jedoch den *Avalanche-Effekt* aus Kapitel 2.3.1 nicht. Um den *Avalanche-Effekt* und die *Geschwindigkeit* in einem Szenario gemeinsam beurteilen zu können, wurde ein Korpus mit 11 Millionen Sätzen benutzt. Es wurde also unter

anderem gemessen, wie viele Sätze mit der jeweiligen Hashfunktion im Speicher verarbeitet werden können, bis der Hash vollgelaufen ist. Dabei wurde ein für *Medusa* entwickeltes Hashing benutzt (siehe Kapitel 2.6.3). In Tabelle 2.4 sind diese Ergebnisse dargestellt.

Hashfunktion	verarbeitete Sätze	% Belegung der Hashtabelle
Adler32	910	0,4
CRC32	893853	91,4
MD2	883719	89,9
MD5	884440	89,9
SHA-160	885677	90,2
SHA-256	890553	90,9
SHA-384	884956	90,1
SHA-512	885357	90,1

Tabelle 2.4: Leistungsfähigkeit verschiedener Hashfunktionen bzgl. ihrer Fähigkeit, zipfverteilte Wort-Kookkurrenzen zu streuen und dadurch möglichst viele Sätze ohne Auslagerung zu zählen

Wie zu erwarten, schneiden die *Kryptographie*-Algorithmen *SHA* und *MD* mit einer durchschnittlichen Auslastung des Hashes von etwa 90% und etwa 885.000 Sätzen (siehe Tabelle 2.4) etwa gleich gut ab. Der *Adler32*-Algorithmus bestätigt nach den Ergebnissen zum *Avalanche-Effekt* aus Tabelle 2.2 ebenfalls sein Verhalten beim Zählen von Kookkurrenzen. Wider Erwarten erreicht der *CRC32*-Algorithmus mit 91,4% die höchste Auslastung und es können dementsprechend auch die meisten Sätze verarbeitet werden. Der Grund dafür scheint in dem Testszenario zum *Avalanche-Effekt* zu liegen. Eine der beiden Wortnummern, die zusammen auftreten sollen, wurde über die Formel aus [New05] berechnet. Die so berechnete Wortnummer entspricht dem *Zipfschen Gesetz*. Die zweite Wortnummer wird auf die gleiche Weise mit der Einschränkung, dass es eine gerade Zahl sein muss, berechnet. Dies hat es ermöglicht, dass das letzte Bit der zweiten Wortnummer dazu genutzt werden kann, um den *Avalanche-Effekt* zu messen, da bei geraden Zahlen dieses Bit immer auf 0 gesetzt ist.

Würde willkürlich ein Bit ausgewählt werden, um es von 0 auf 1 zu setzen, so lässt sich an dieser Stelle auf Basis der Daten aus Tabelle 2.4 nur vermuten, dass auch für den *CRC32*-Algorithmus ein besserer *Avalanche-Effekt* gemessen werden kann.

Bei der Wahl der zu benutzenden Hashfunktion fällt die Entscheidung auf den *CRC32*-Algorithmus. Es hat sich gezeigt, dass dieser Algorithmus einer-

seits zu den schnellsten gehört und andererseits am besten die Daten auf eine Hashtabelle uniform streuen kann. Die *Kryptologie*-Algorithmen funktionieren letztlich genau so gut wie der *CRC32*-Algorihtmus. Sie benötigen aber allesamt mehr als das Dreifache der Zeit. Damit sind sie für das Zählen von Kookkurrenzen nicht geeignet. Ein Grund dafür liegt einerseits in der Komplexität dieser Algorithmen. Andererseits werden „Hashwerte“ erzeugt, die deutlich länger sind als die vier Byte des *CRC32*-Algorithmus (vgl. Tabelle 2.3), so dass allein dies bereits mehr Zeit kostet.

2.4 Adresskollisionen

Ein Unterscheidungskriterium von Hashverfahren ist die Möglichkeit von Adresskollisonen. Eine Adresskollision entsteht, wenn die Hashfunktion h für zwei verschiedene Schüssel $k, k' \in \mathcal{K}$ die gleiche Hashadresse $h(k) = h'(k)$ berechnet.

Ein *perfektes Hashverfahren* produziert keine Adresskollisionen. Bei diesen Verfahren wird sichergestellt, dass jedem Schlüssel eine eigene Hashadresse zugewiesen wird. Dies funktioniert jedoch nur bei sehr kleinen Mengen von Schlüsseln. In den meisten Fällen jedoch ist die Menge der möglichen Schlüssel deutlich größer als der Hashbereich der Hashtabelle. Somit ist für größere Probleme ein *perfektes Hashing* nicht mehr möglich. In diesen Fällen wird von *offenen Hashverfahren* gesprochen. Bei diesen Verfahren sind Adresskollisionen jedoch nicht zu vemeiden.

Im Rahmen von *Medusa* werden beide Formen des Hashings benutzt. Ein *perfektes Hashverfahren* wird für die häufigsten Wörter eines Korpus benutzt. Dabei wird davon ausgegangen, dass zwischen diesen Wörtern eine hohe Kookkurrenz-Dichte herrscht. Für die seltenen und mittelfrequenten Wörter wird ein *offenes Hashverfahren* verwendet.

Um eine Entscheidung treffen zu können, in welchem Hash die Kookkurrenz gezählt werden soll, wird intern mit Wortnummern gearbeitet. Die Wortnummern werden dabei nach der Frequenz des Wortes vergeben. Das frequenteste Wort hat somit die Wortnummer 1.

Um diese Entscheidung treffen zu können, muss lediglich überprüft werden, ob die beiden Wortnummern einer Kookkurrenz unter einer vorher definierten Schwelle von beispielsweise 5.000 liegen. Wenn dies der Fall ist, dann wird die Kookkurrenz im *Array-Hash* mit einem *perfekten Hashverfahren* gezählt werden. Andernfalls wird die Kookkurrenz im *RAM-Hash* mittels eines *offenen Hashverfahrens* berechnet.

Abbildung 2.3 stellt den Ablauf des Zählens von Kookkurrenzen innerhalb von *Medusa* dar. Dabei wird auf einem Eingabetext ein Kookkurrenzfilter,

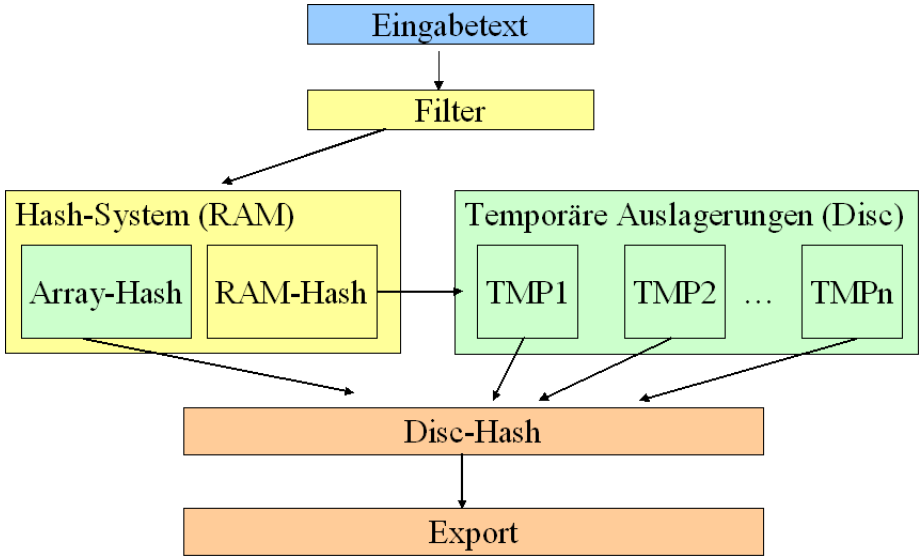

Abbildung 2.3: *Medusa*'s Hashingsystem, bestehend aus drei Hashes: *Array*-, *RAM*- und *DISC*-Hash

wie beispielsweise der im Rahmen dieser Arbeit oft benutzte Filter der Satzkookkurrenzen, angewendet. Dieser Filter übergibt das *Kookkurrenz*-Token an das Hashsystem, welches wie eben beschrieben darüber entscheidet in welchem Hash die Kookkurrenz gezählt werden soll. Läuft der *RAM*-Hash voll, dann wird dieser auf die Festplatte ausgelagert und anschließend geleert. Nachdem der gesamte Text verarbeitet worden ist, werden der *Array*-Hash sowie die ausgelagerten *RAM*-Hashes zu einem großen *Disc*-Hash zusammengeführt. Dieser ermöglicht das persistente Ablagern von Kookkurrenzen, so dass bei neuem Textmaterial nur noch die neuen Sätze verarbeitet werden müssen und so eine weitere Verbesserung der Performanz sichergestellt ist.

2.5 Wahl der *Modulo*-Operation

In Kapitel 2.3.2 wurde gezeigt, dass mittels einer Hashfunktion eine zipfverteilte Menge $\mathcal{K}$ von Schlüsseln so auf einen Hash verteilt werden kann, dass dieser effizient genutzt und zu über 90% ausgelastet werden kann. Die numerischen Werte, die ein solcher Algorithmus zurückliefert, sind dabei wesentlich größer als es Hashwerte in der Hashtabelle gibt. Um diese an die Größe der Hashtabelle s anzupassen, wird in der Regel die *Modulo*-Operation

$$h(k) = crc32(k) \bmod s \tag{2.3}$$

mit $(k_1, k_2, \ldots, k_n) = k \in \mathcal{K} \subseteq T^n$ benutzt.

Bei der Wahl von s wird empfohlen (vgl. [Knu98] bzw. [OW96]), dass es sich um eine Primzahl handeln sollte. Wird s gerade gewählt, dann ist $h(k)$ gerade, wenn $crc32(k)$ eine gerade Zahl zurückliefert. Entsprechendes gilt auch für ungerade Ergebnisse, die durch $crc32(k)$ berechnet werden. Da die $crc32(k)$-Funktion die Keys k sehr gut auf gerade und ungerade Zahlen verteilt, kann diese Forderung vernachlässigt werden.

Ferner wird oftmals zusätzlich gefordert, dass s so zu wählen ist, dass es möglichst in gleichem Abstand zwischen zwei Zahlen der Form 2^n liegt (vgl. [Knu98]). Würden die Hashwerte einer Hashtabelle wie in der folgenden Formel berechnet werden

$$h(k) = crc32(k) \bmod 2^t, \tag{2.4}$$

dann würden lediglich die letzten $\log_2 2^t = t$ Bits der Funktion $crc32(k)$ für die Berechnung des Hashwertes eine Rolle spielen. Die restlichen $(32 - t)$ Bits bleiben unberücksichtigt. Diese Kritik ist insofern berechtigt, als dass einige Funktionen wie die *Multiplikation* oder die *Addition* von zwei Wortnummern einen sehr schlechten *Avalanche-Effekt* haben (vgl. Tabelle 2.2). Würden auf diese Weise nur die letzten t Bits für die *Modulo*-Operation benutzt werden, so könnten relevante Bitänderungen verloren gehen.

Angesichts der guten *Avalanche-Effekte* für anspruchsvollere Algorithmen ist dies jedoch nicht mehr notwendig, da jede Bitänderung etwa 50% aller anderen Bits über den gesamten Bereich dieser Funktion ändern kann. Auf diese Weise kann die komplizierte und langsame *Modulo*-Operation durch eine bitweise *UND*-Verknüpfung ersetzt werden. Im Falle der $crc32(k)$-Funktion bedeutet dies, dass eine Bitmaske der Länge 32 erstellt wird, in welcher die ersten $32 - t$ Bits auf 0 und die letzten t auf 1 gesetzt sind. Dann kann wie in der folgenden Formel

$$\begin{pmatrix} crc32_{31}(k) \\ \vdots \\ crc32_{t+1}(k) \\ crc32_t(k) \\ crc32_{t-1}(k) \\ \vdots \\ crc32_0(k) \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ crc32_t(k) \\ crc32_{t-1}(k) \\ \vdots \\ crc32_0(k) \end{pmatrix}.$$

der zu benutzende Hashwert ohne eine *Modulo*-Operation berechnet werden. Auslastungen von über 90% zeigen, dass bei einer guten Hashfunktion auch eine Zweierpotenz zur Moduloberechnung genutzt werden kann. Dies deckt

sich auch mit Ergebnissen von Zobel et al. (vgl. [ZHW01]), die Wörter in einem Hash gezählt haben. Tests haben weiterhin ergeben, dass diese Form der *Modulo*-Berechnung etwa doppelt so schnell ist, wie das Berechnen des Restes in der Programmiersprache Java. Für das Berechnen von Kookkurrenzen bedeutet dies, dass ungefähr 10.000 Sätze/Minute mehr verarbeitet werden können.

2.6 Hashen mit zwei Hashfunktionen

Bei einem Studium der gegenwärtigen Fachliteratur zum Thema Hashing, können viele wenig bekannte Hashansätze kennengelernt werden. Ein Trend, der bei den Vorbereitungen zur Entwicklung von *Medusa* ausgemacht werden konnte, ist das Hashen mit zwei Hashfunktionen. In Kapitel 2.3.1 wurde gezeigt, dass Hashfunktionen selbst bei ungleichmäßig verteilten Eingabedaten diese sehr gut über den Hashbereich verteilen. Dennoch steigt die Zahl der Adresskollisionen bei zunehmender Auslastung des Hashes stark an.

In diesem Sinn soll in diesem Unterkapitel auf drei Ansätze des Hashens mit zwei Hashfunktionen eingegangen werden. Das wohl bekannteste Verfahren dazu ist das sogenannte *Double Hashing*. Ein weiteres Verfahren, das allerdings nur kurz skizziert werden soll, ist das *Cuckoo Hashing*. Das *Cuckoo Hashing* dient auch als Grundlage für das Hashing, welches in Medusa benutzt wird. Diese Form des Hashens mit zwei Hashfunktion soll als letzte Möglichkeit vorgestellt werden.

2.6.1 Double Hashing

Der Grundgedanke des *Double Hashings* ist, die marginalen Ungleichverteilungen (siehe Kapitel 2.3.1 und Tabelle 2.2) der einen Hashfunktionen durch die Streuung einer zweiten Hashfunktion auszugleichen. Dabei wird die zweite Hashfunktion als Sondierungsfunktion benutzt. Beide Hashfunktionen sollten so gewählt werden, dass sie möglichst unabhängig von einander sind.

$$h(k) = (f_1(k) - j * f_2(k)) \bmod s \quad mit \quad k \in \mathcal{K} \subseteq T^n \tag{2.5}$$

Dabei sind f_1 und f_2 die beiden Hashfunktionen und $j = 0, 1, 2, \ldots$ ist ein Zähler dafür, wie viele Sonderierungen für den Schlüssel $k \in \mathcal{K}$ bereits gemacht werden mußten. Auf diese Weise läßt sich die Anzahl der Sonderierungen beim Einfügen eines *Schüssel-Wert*-Paares selbst bei Auslastungen von mehr als 80% auf unter 2,5 im Durschnitt halten (vgl. [OW96]). Eine Eigenschaft für diese Form des Hashens mit zwei Hashfunktionen ist, dass

beide Hashfunktionen auf dem gleichen Hashbereich arbeiten. Es könnte also von beiden Hashfunktionen aus jeder Hashwert berechnet werden.

2.6.2 Cuckoo Hashing

Das *Cuckoo Hashing* (vgl. [PR04]) teilt den Hashbereich in zwei gleich große Teile. Jeder dieser halbierten Blöcke wird von einer anderen Hashfunktion verwaltet. Jede Hashfunktion verwaltet also nur die Hälfte des gesamten Hashbereichs. Eine Adressierung außerhalb des zugewiesenen Hashbereichs ist nicht möglich.

Beim Einfügen eines *Schüssel-Wert*-Paares wird zuerst ein Hashwert im ersten Hashbereich mittels der ersten Hashfunktion $f_1(k)$ berechnet. Sollte dort bereits ein anderes *Schüssel-Wert*-Paar gespeichert sein, dann wird die zweite Hashfunktion $f_2(k)$ benutzt, um einen Speicherplatz im zweiten Hashbereich zu berechnen. Sollte auch dort bereits ein *Schüssel-Wert*-Paar gespeichert sein, dann wird der dort gespeicherte Werte in den ersten Hashbereich kopiert. Dies kann aber wiederum ein weiteres Kopieren nach sich ziehen. Nachdem nun im zweiten Hashbereich die berechnete Hashadresse wieder frei ist, kann das neu einzufügende *Schüssel-Wert*-Paar dort platziert werden. Für Details sei auf die zahlreichen Paper von Pagh wie beispielsweise [PR04] verwiesen.

Diese Form des Hashens mit zwei Hashfunktionen hat gegenüber dem *Double Hashing* einen Vorteil und einen Nachteil. Der Vorteil ist, dass temporär die zuletzt eingefügten *Schüssel-Wert*-Paare mit $O(1)$ erreichbar sind, wenn sie im ersten Hashbereich und mit $O(2)$, wenn sie im zweiten Hashbereich abgelegt worden sind. Da sich Wörter mitunter sehr ungleichmäßig über einen Korpus verteilen, treten auch die Kookkurrenzen ungleichmäßig auf. Solche lokal im Korpus auftretenden Kookkurrenzen könnten somit selbst bei einem Hash mit einer hohen Auslastung schnell erreichbar sein. Im Gegenzug werden Kookkurrenzen nach „hinten“ verschoben, die vielleicht sehr früh aufgetreten sind und sich über den gesamten Korpus verteilen. Der große Nachteil ist das viele Kopieren der *Schüssel-Wert*-Paare. Dies beansprucht zusätzliche Zeit. Die Ergebnisse von Pagh (vgl. [PR04]) zeigen, dass diese Form des Hashens am Ende deutlich langsamer ist als ein normales Hashing mit *linearer Sondierung*. Nach Pagh[6] ist die *lineare Sondierung* immer schneller als das *Double Hashing*. Somit wurde bei seinen Forschungen das *Double Hashing* nicht weiter betrachtet.

[6]Pagh ist nicht nur Entwickler des *Cuckoo Hashings*, sondern auch in der Forschung im Bereich des Hashings aktiv. So sei an dieser Stelle auf [Pag99], [sP02] und [Pag02] verwiesen, die wichtige Grundlagen des im Rahmen dieser Arbeit vorgestellten Hash-Ansatzes zur Verfügung stellen.

2.6.3 Medusa's Hashing

Auch wenn das *Cuckoo Hashing* langsamer ist als ein normales Hashing mit linearer Sondierung, so ist der Grundgedanke des Teilens eines globalen Hashraumes in mehrere lokale Hashbereiche interessant. Im Rahmen der Entwicklung von *Medusa* wurde dieser Gedanke weitergedacht und ein zweistufiges Hashing als *Divide & Conquer*-Strategie entwickelt.

Als eine *Divide & Conquer*-Strategie wird ein Algorithmus bezeichnet, der zuerst ein großes und mit nur hohem Zeitaufwand berechenbares Problem möglichst sinnvoll in mehrere Teile zerlegt. Danach werden die kleinen Teilprobleme jeweils lokal gelöst. Als letztes werden die Teilergebnisse wieder zu einem großen Ergebnis zusammengesetzt. Als Sondierungsfunktion im Falle einer Kollision wird das *lineare Sondieren* benutzt.

Bezogen auf das Hashing bedeutet dies, dass der Hashbereich s in mehrere *Buckets* B geteilt wird. Ein Bucket ist dabei ein Fragment des gesamten Hashbereichs. Jeder Bucket besitzt somit $z = \frac{s}{B}$ Hashwerte. s wird dabei so gewählt, dass $s \bmod B = 0$ gilt. Unter Ausnutzung der Ergebnisse aus Kapitel 2.5 wird die Anzahl der Buckets B als eine Zweierpotenz gewählt. Die erste Hashfunktion $f_1(k)$ verteilt die *Schüssel-Wert*-Paare auf die einzelnen Buckets. Dies entspricht dem Teilen bei der *Divide & Conquer*-Strategie. Anschließend wird innerhalb des Buckets mit einer zweiten Hashfunktion $f_2(k)$ nur noch auf einem kleinen Fragment des Hashbereichs das Problem gelöst. Das Zusammensetzen geschieht implizit, da der Hashbereich nie geteilt war. Die Berechnung des Hashwertes lässt sich schließlich wie in Formel 2.6 zusammenfassen.

$$h(k) = (f_1(k) \bmod B) * z + (f_2(k) \bmod b) \tag{2.6}$$

Wobei B für die Anzahl der Buckets, z für die Größe des Buckets (in Byte) und b für die Anzahl der Hashadressen innerhalb eines Buckets stehen.

In Tabelle 2.5 werden einige Ergebnisse mit unterschiedlichen Konfigurationen bezüglich des Verhältnisses Buckets und Hashwerte pro Bucket abgebildet. Bei diesem Test wurde von einem RAM-Hash der Größe K_{RAM}

$$K_{RAM} = B * S_B \tag{2.7}$$

ausgegangen. B steht dabei für die Anzahl der Buckets. S_B steht für die Größe eines Buckets in Byte. Wobei sich S_B wie in Formel 2.8 berechnet.

$$S_B = z * (\sum_{k=1}^{n} s_{WortNr,k} + s_{RAMHash,Frequenz}) + \lceil \log_2 z \rceil \tag{2.8}$$

z gibt an, wie viele Hashwerte pro Bucket zur Verfügung stehen.

$\sum_{k=1}^{n} s_{WortNr,k} + s_{RAMHash,Frequenz}$ berechnet die Anzahl der Bytes, die für einen Hashwert zur Verfügung stehen müssen. Für Kookkurrenzen zweiter Ordnung ($n = 2$) gilt somit die Vereinfachung $s_{WortNr,1} + s_{WortNr,2} + s_{Frequenz}$. Der letzte Term dieser Formel $\lceil \log_2 z \rceil$ berechnet die Anzahl der Bytes, welche zur Speicherung der Anzahl der Datensätze benutzt werden, die in dem jeweiligen Bucket gespeichert werden sollen. Dies läßt sich auch wie in Formel 2.9 vereinfachen.

$$\lceil \log_2 z \rceil = \begin{cases} 1 & \text{für z} \leq 255 \\ 2 & \text{für } 256 \geq z \leq 65535 \\ 3 & \text{für } 65536 \geq z \leq 16777215 \\ 4 & \text{für z} \geq 16777216 \end{cases} \tag{2.9}$$

Aus den Formeln 2.7 und 2.8 ergibt sich vielmehr die Formel 2.10.

$$K_{RAM} = B * (z * (\sum_{k=1}^{n} s_{WortNr,k} + s_{RAMHash,Frequenz}) + \lceil \log_2 z \rceil) \tag{2.10}$$

Für die folgenden Tests wurde K_{RAM} so gewählt, dass möglichst genau 850 MB Arbeitsspeicher für den RAM-Hash benutzt werden. Es war nicht immer möglich, die 850 MB genau so zu teilen, dass die Formel 2.7 exakt erfüllt wird. In diesem Fall wurde der nächst mögliche kleinere Wert benutzt. In der Startkonfiguration wurde von $B = 16.777.216$ ausgegangen. Bei einer größeren Wahl von B werden selbst geringe Ungleichverteilungen der Hashfunktion spürbar, so dass bereits sehr früh ausgelagert werden muss. In jeder weiteren Konfiguration wurde die Anzahl der Buckets halbiert (siehe Tabelle 2.5). Die Größe des Buckets wurde dementsprechend etwa verdoppelt. In der Tabelle 2.5 sind die Ergebnisse bis zu einer Bucketanzahl von 2.048 abgebildet. Die zweite Spalte der Tabelle 2.5 enthält die Anzahl der Hashwerte pro Bucket. Um diese zu berechnen wurde die Formel 2.10 nach z umgestellt (Formel 2.11). Zur Vereinfachung von $\lceil \log_2 z \rceil$ wurde eine Lookup-Tabelle wie in Formel 2.9 benutzt.

$$z = \left\lfloor \frac{\frac{K_{RAM}}{B} - \lceil \log_2 z \rceil}{(\sum_{k=1}^{n} s_{WortNr,k} + s_{Frequenz})} \right\rfloor \tag{2.11}$$

Die Tabelle 2.5 enthält Spalten für

- die *Anzahl der Buckets*, die, wie oben erklärt, bestimmt worden sind,
- die *Anzahl der Hashwerte pro Bucket*, die nach Formel 2.11 berechnet worden sind,

- die Anzahl der Buckets, die *leer* geblieben sind,
- die *Anzahl der benutzten Hashwerte*, in dem Bucket, welches am wenigsten belegt ist,
- die *Anzahl der Sätze*, die verarbeitet werden konnten, bis ein Bucket vollgelaufen war,
- die *benötigte Zeit* für den gesamten Prozess,
- die *Anzahl der verarbeiteten Sätze* pro Sekunde,
- die *durchschnittliche Anzahl* der Zugriffe, um eine Kookkurrenz einzufügen bzw. die Frequenz einer bereits gespeicherten Kookkurrenz um 1 zu erhöhen,
- die *Anzahl der Kookkurrenzen*, die während dieses Prozesses verarbeitet werden konnten und
- die *durchschnittliche Auslastung* des RAM-Hashes.

Aus Tabelle 2.5 kann entnommen werden, dass ein Hash eine größere Auslastung erreichen kann, wenn die Anzahl der Buckets so gering wie möglich gewählt wird. Dies geht jedoch auf Kosten der benötigten Zeit. Für $B = 1$ wurde der Test nach mehreren Stunden abgebrochen, da Kookkurrenzen nicht mehr spürbar gezählt wurden (vgl. Durchsatz für diese Konfiguration). Diese Konfiguration entspricht einem normalen Hashing mit linearer Sondierung. Andererseits wäre es möglich gewesen, dass die rund 111 Millionen zur Verfügung stehenden Hashwerte auch belegt werden könnten. Bereits bei $B = 2048$ ist die Rechenzeit mit rund 25 Minuten deutlich besser. Die Hashauslastung liegt mit 98,85% immer noch sehr stark am Maximum. In dieser Konfiguration enthält das Bucket, welches die wenigsten Kookkurrenzen gespeichert hat, 53.059 *Schüssel-Wert*-Paare. Da bei dieser Konfiguration jedes Bucket 54.408 Hashadressen besitzt, ergibt sich für das am schlechtesten ausgelasteten Bucket immer noch eine Auslastung von $\frac{53.059}{54.408} = 0,975$. Die Rechenzeit lässt sich weiter reduzieren, indem die Anzahl der Buckets erhöht wird.

Die besten Resultate konnten mit einer Belegung von $B = 65.536$ und $B = 32.768$ erreicht werden. In beiden Fällen wurde der Hash zu über 90% bei etwa zwei Zugriffen im Schnitt ausgelastet. Dementsprechend konnten in beiden Fällen mehr als 100 Millionen Kookkurrenzen im Speicher gezählt werden. Die minimale Auslastung eines Buckets beträgt 83% bzw. 87,9%. In beiden Fällen konnten etwa 900.000 Sätze verarbeitet werden. Damit konnten zwar rund 50.000 Sätze weniger als bei einer Belegung von $B = 2048$ gezählt

Anzahl der Buckets B	Hashwerte pro Bucket z	Anzahl der leeren Buckets	minimale Bucket-belegung	verarbeitete Sätze	benötigte Zeit in Sek.	Durchsatz in Sätze pro Sek.	durchschn. Anzahl der Zugriffe	gezählte Kookk.	durchschn. Auslastung des Hashes
16.777.216	6	10.384.794	0	25.845	35	738	2,01	7.757.437	0,0771
8.388.608	13	404.673	0	90.454	116	779	2,05	22.801.317	0,2091
4.194.304	26	31	0	190.225	238	799	2,10	42.010.270	0,3852
2.097.152	53	0	9	288.520	346	833	2,18	58.172.916	0,5234
1.048.576	106	0	33	699.670	424	1.650	2,26	68.697.335	0,6181
524.288	212	0	103	766.540	513	1.494	2,42	80.477.089	0,7240
262.144	425	0	278	838.193	618	1.356	3,08	92.607.075	0,8312
131.072	850	0	642	865.912	665	1.302	3,30	97.235.597	0,8728
65.536	1.700	0	1.411	893.853	714	1.251	3,61	101.822.323	0,9139
32.768	3.400	0	2.987	912.754	773	1.180	3,86	104.900.253	0,9416
16.384	6.801	0	6.176	924.731	826	1.119	4,57	106.801.133	0,9585
8.192	13.602	0	12.859	935.829	965	969	6,50	108.571.662	0,9744
4.096	27.204	0	26.116	941.571	1159	812	9,56	109.509.629	0,9828
2.048	54.408	0	53.059	945.530	1509	626	17,43	110.147.545	0,9885
1	111.427.584	0	111.427.584	ca. 950.000	> 42.183	< 22	> 674	111.427.584	1

Tabelle 2.5: *Medusa*'s Hashing mit „konstantem“ globalen Hashbereich s und unterschiedlicher Fragmentierungsstärke B

werden, aber bei einer durchschnittlichen Verarbeitungszeit von etwa 50.000 Sätzen pro Minute, kann dies nach dem Auslagern des RAM-Hashes auf die Festplatte in etwa einer Minute erledigt werden. Damit sind Belegungen von $B = 32.768$ und $B = 65536$ mehr als zehn Minuten schneller bei der Verarbeitung von etwa 950.000 Sätzen.

Wird die Anzahl der Buckets B über 65.536 weiter erhöht, so kann auch die Anzahl der Zugriffe weiter gesenkt werden. Jedoch werden auch deutlich weniger Sätze verarbeitet, bevor der *RAM-Hash* vollläuft und auf die Festplatte ausgelagert werden muss. Dem liegt zugrunde, dass durch das Verdoppeln von B die Anzahl der Hashwerte z bei konstantem s halbiert werden. Durch die geringere Anzahl der Hashwerte pro Bucket werden „Ungleichverteilungen“ der Hashfunktion deutlich spürbar.

2.7 Kombination von perfektem und offenem Hashing

In Kapitel 2.4 wurde der Unterschied zwischen *perfektem* und *offenem* Hashing erklärt. Es wurde gezeigt, dass es für das Zählen von Kookkurrenzen nicht möglich ist, ein *perfektes* Hashing einzusetzen. Weiterhin wäre es auch nicht sinnvoll für jedes $k \in \mathcal{K} \subseteq T^n$ einen Speicherplatz in einem n-dimensionalen Raum freizuhalten. Für $n = 2$ kann gerade einmal jede 20.000ste Kookkurrenz aus T^n auch gefunden werden. Dennoch macht es Sinn, die Kollisionsfreiheit des perfekten Hashings auszunutzen.

Im Rahmen von *Medusa* wird zusätzlich zu dem in Kapitel 2.6.3 vorgestellten Hashing ein zweiter Hashbreich eingesetzt. In diesem Hashbereich werden Kookkurrenzen zwischen den häufigsten Wörtern gezählt. Dieser Hashbereich kann als eine zweidimensionale *Matrix* aufgefasst werden. Dabei werden die Wortnummern implizit als Index für die Matrix gespeichert, so dass pro Kookkurrenz nur noch die Anzahl der Bytes für die Frequenzzählung allokiert werden muss. Dies reduziert den Speicherbedarf für eine Kookkurrenz um etwa 66%.

Für die Wahl der Größe des *Matrix*-Hashes wurden einige Tests gemacht. Für diese Tests wurde ein $n \times n$-*Matrix-Hash* benutzt. Dabei wurde der Arbeitsspeicher von einem Gigabyte mit $n = 17.000$ bestmöglich ausgenutzt.

Im ersten Test, für den an dieser Stelle die Ergebnisse vorgestellt werden sollen, wurde der Anstieg der gezählten *Kookkurrenz-Tokens* bestimmt, wenn für den *Matrix*-Hash pro Dimension der Hashbereich um eins erhöht wird. In Abbildung 2.4 ist der entsprechende Plot abgebildet.

Aus Abbildung 2.4 kann entnommen werden, dass der maximale Nutzen,

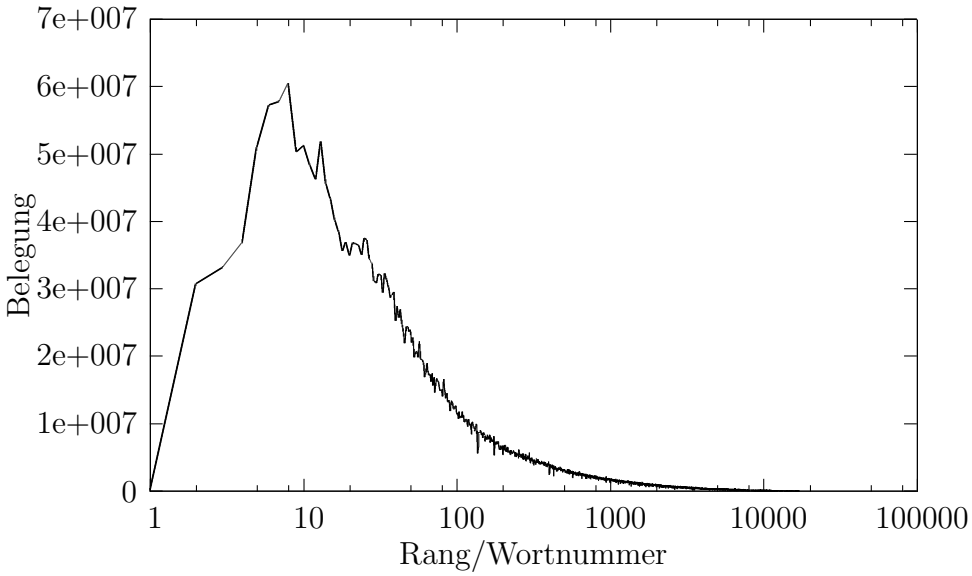

Abbildung 2.4: Anzahl, der zusätzlich gezählten *Kookkurrenz-Tokens* für die Spalte i des *Matrix-Hashes*, wenn der *Matrix-Hash* um die Spalte i vergrößert wurde

der durch das Erweitern des *Matrix-Hashes* erreicht werden kann, bei 10 liegt. Bei einem *Matrix-Hash* der Größe 100×100 würden in der Spalte 100 und in der Zeile 100 immer noch rund eine Million *Kookkurrenz-Tokens* gezählt werden können. Aufgrund der hohen Dichte, auf die noch später eingegangen wird, empfiehlt es sich, einen 5000×5000- bis 10000×10000-*Matrix-Hash* zu benutzen, auch wenn, wie in Abbildung 2.4 deutlich sichtbar, die Anzahl der gezählten *Kookkurrenz-Tokens* mit rund 50.000 sehr gering ist.

Ein weiterer Test betraf die Belegung des *Matrix-Hashes* unter bestimmten Größen. Dazu wurde der gleiche Testaufbau wie im vorigen Beispiel benutzt. Danach wurde für $n = 1$ beginnend die Anzahl der belegten Hashwerte h_b und der unbelegten Hashwerte h_u bestimmt. Schließlich wurde die Belegung b durch $b = \frac{h_b}{h_b + h_u}$ berechnet. Der Verlauf von b wurde in Abbildung 2.5 für unterschiedliche Größen des *Matrix-Hashes* abgebildet. Abbildung 2.4 ließ vermuten, dass für das Zählen von Kookkurrenzen der *Matrix-Hash* eher klein gewählt werden sollte. Aus Abbildung 2.5 kann entnommen werden, dass ein 5000×5000-*Matrix-Hash* eine Auslastung von 97,9% und ein 10000×10000-*Matrix-Hash* immerhin noch eine Auslastung von rund 80% erreicht. Dies ist insofern von besonderem Interesse, da der Hash relativ voll ist. Im Vergleich zu den *offenen Hash-Verfahren*, die bei solchen Auslastungen bereits sehr viele Kollisionen produzieren, kann an dieser Stelle weiterhin

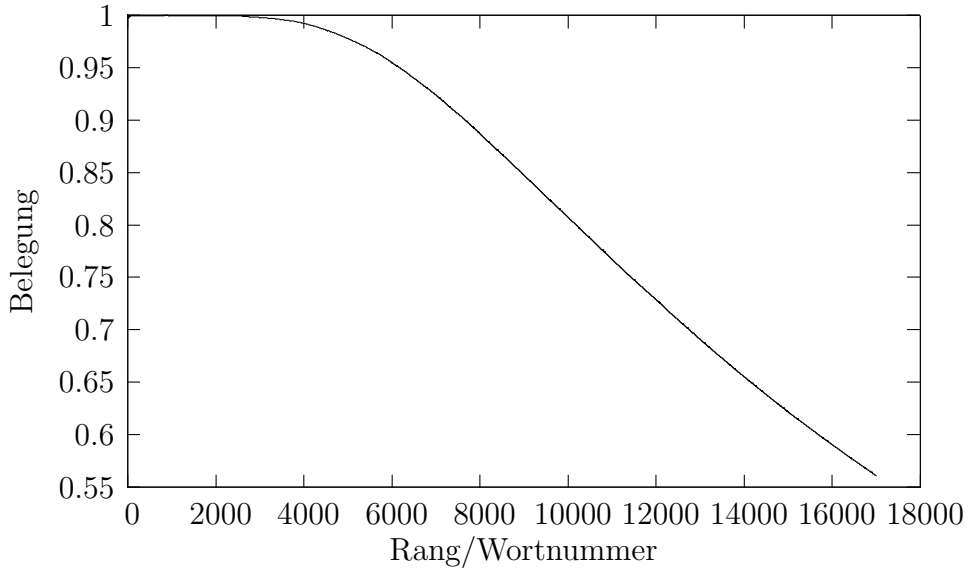

Abbildung 2.5: Belegungsgrad, des *Matrix-Hashes* mit steigender Größe

mit optimal genau einem Zugriff gezählt werden.

Um das Zusammenspiel von *Matrix-* und *RAM*-Hash zu testen, wurde ein 11 Millionen Sätze großer Korpus benutzt. Somit galt es wieder, wie in den Tests zu Tabelle 2.5, verschiedene Konfigurationen miteinander zu vergleichen. Dabei standen vor allem zwei Kriterien im Mittelpunkt. Einerseits sollte überprüft werden, inwiefern durch die Wahl eines größeren bzw. kleineren *Matrix-Hashes* mehr Sätze ohne Auslagern des *RAM-Hashes* verarbeitet werden können. Andererseits kann in einem *perfekten* Hash wie dem *Matrix-Hash* mit genau einem Zugriff pro Kookkurrenz gezählt werden. Bei den *offenen Hashverfahren* ist dies nicht der Fall. Deswegen sollte auch überprüft werden, inwiefern sich die *durchschnittliche Anzahl* der Zugriffe auf das *Gesamtsystem* verändert, wenn die Größe des *Matrix-Hash* verändert wird.

Für dieses Szenario wurde, wie in Kapitel 2.6.3, von einer vorgegebenen Größe K_{Gesamt} des Hashes von rund 850 MB ausgegangen. Dieser berechnet sich nach Formel 2.12.

$$K_{Gesamt} = K_{Array} + K_{RAM} \tag{2.12}$$

K_{Array} berechnet sich dabei für die n frequentesten Wörter wie in Formel 2.13.

$$K_{Array} = n^2 * s_{Array,Frequenz} \tag{2.13}$$

In [Bie06a] weist Biemann auf die Symmetrie mancher Koookkurrenzen hin.

So werden bei Satzkookkurrenzen die *Kookkurrenz-Tokens* $(wort_1, wort_2)$ und $(wort_2, wort_1)$ gefunden. Diese Symmetrie kann ausgenutzt werden, so dass nur eines der beiden Tokens gespeichert werden muss. Das andere Token wird *post-precessing* erzeugt. Dies spart in beiden Hashes etwa die Hälfte des Platzes bzw. es kann etwa doppelt so viel Text ohne Auslagerung verarbeitet werden. Für den *Matrix-Hash* muss dazu jedoch aus der zweidimensionalen Matrix eine eindimensionale gemacht werden. Für die Adressierung dieses neuen Hashes schlägt Biemann die Formel 2.14 vor.

$$a = \frac{wort_1^2 - wort_1}{2} + wort_2 \tag{2.14}$$

Bei den folgenden Tests wurden Satzkookkurrenzen mit der eben vorgestellten Optimierung berechnet. Damit berechnet sich die Größe des modifizierten *Matrix-Hashes* (Aufgrunddessen, dass es sich nicht mehr um eine Matrix handelt, wird dieser Hash nun *Array-Hash* genannt.), nach der Formel 2.15.

$$\begin{aligned} K_{Array} &= (n + n - 1 + \ldots + 2 + 1) * s_{Array,Frequenz} \\ &= (\sum_{i=1}^{n} i) * s_{Array,Frequenz} \\ &= \frac{n}{2} * (n + 1) * s_{Array,Frequenz} \end{aligned} \tag{2.15}$$

Ausgehend von einer festen Gesamtgröße K_{Gesamt} des „Hashesystems“ (vgl. Formel 2.12) wurde die Größe des *Array-Hashes* in 500er Schritten, bei 0 beginnend, erhöht. Die Größe des *RAM-Hashes* wurde in Anlehnung an Formel 2.12 angepasst. Bei der Konfiguration des *RAM-Hashes* wurde von einer festen Anzahl der Buckets $B = 65.536$ ausgegangen (vgl. Ergebnisse aus Kapitel 2.6.3 bzw. Tabelle 2.5). Die Anzahl der Hashwerte pro Bucket wurde dementsprechend angepasst. Aus den Formeln 2.12, 2.15 und 2.11 kann schließlich durch Einsetzen die Formel 2.16 hergeleitet werden, die den gesamten Speicherverbrauch beider Hashes angibt.

$$K = \frac{n}{2} * (n+1) * s_{Array,Freq} + B * (z * (\sum_{k=1}^{n} s_{WortNr,k} + s_{RAMHash,Freq}) + \lceil \log_2 z \rceil) \tag{2.16}$$

Da K_{Gesamt} konstant ist und n kontrolliert in 500er Schritten erhöht wird, muss die Formel nach z umgestellt werden, um bei einem gewählten n die Anzahl der Hashwerte der Buckets des *RAM-Hashes* so zu bestimmen, dass K_{Gesamt} auch „relativ“ konstant bleibt. Die Formel 2.17 stellt diese Umstellung dar.

$$z = \left\lfloor \frac{K_{Gesamt} - B * \lceil \log_2 z \rceil - \frac{n}{2} * (n + 1) * s_{Array,Frequenz}}{B * \sum_{k=1}^{n} s_{WortNr,k} + s_{RAMHash,Frequenz}} \right\rfloor \tag{2.17}$$

In Tabelle 2.6 sind die Ergebnisse dieses Tests abgebildet. Diese Tabelle enthält Spalten für

- die *Größe des Array-Hashes*, welcher von 0 beginnend um jeweils 500 erhöht wurde,
- die *Anzahl der Hashwerte* pro Bucket, die nach Formel 2.17 berechnet worden sind,
- die *Anzahl der Sätze*, die bis zum ersten Auslagern verarbeitet werden konnten,
- die Anzahl der *Zugriffe auf den Array-Hash* (in Millionen),
- der Anteil der *Zugriffe auf den Array-Hash* bezogen auf die Gesamtanzahl (*Array-Hash* und *RAM-Hash*) aller Zugriffe,
- die Anzahl der *Zugriffe auf den RAM-Hash* (in Millionen),
- der Anteil der *Zugriffe auf den RAM-Hash* bezogen auf die Gesamtanzahl aller Zugriffe,
- die Gesamtanzahl aller Zugriffe (in Millionen),
- die Anzahl der Kollisionen im *RAM-Hash* (in Millionen),
- die *durchschnittliche Anzahl* der Zugriffe,
- die *benötigte Zeit*, um die Satzkookkurrenzen für die in der dritten Spalte stehende Anzahl von Sätzen zu berechnen sowie
- der Durchsatz an Kookkurrenzen, die in einer Sekunde verarbeitet werden konnten.

Die erste Erkenntnis aus Tabelle 2.6 ist, dass durch das Vergrößern des *Array-Hashes* nicht entscheidend mehr Sätze verarbeitet werden können. Wird kein *Array-Hash* benutzt, dann können nach Tabelle 2.6 rund 900.000 Sätze ohne ein Auslagern des *RAM-Hashes* auf die Festplatte verarbeitet werden. Im Maximum können bei einem *Array-Hash*, welcher die Kookkurrenzen für die 5500 bzw. 6000 häufigsten Wörter speichert, rund 914.000 Sätze verarbeitet werden. Die Hinzunahme eines *perfekten* Hashes kann also nicht dazu genutzt werden, um deutlich mehr Sätze zu verarbeiten.

Eine weitere Beobachtung, die der Tabelle 2.6 entnommen werden kann, ist, dass bei einem *Array-Hash* für die Kookkurrenzen der 2.500 häufigsten Wörter bereits mehr als die Hälfte aller *Kookkurrenz-Tokens* gezählt werden

Größe Array-hashes	RAM-Hash Haswerte Bucket	ver-arbeitete Sätze	Zugriffe Array-Hash in Mio.	Anteil der Zugriffe	Zugriffe RAM-Hash in Mio.	Anteil der Zugriffe	Summe der Zugriffe in Mio.	Anzahl der Koll. in Mio.	⌀ Anzahl der Zugriffe	benötigte Zeit in Sek.	Durchsatz in Kookk. pro Sek.
0	1700	893.853	0	0,000	219	1,000	219	570	3,61	720	303.472
500	1700	893.853	144	0,329	293	0,671	437	560	2,28	621	703.478
1.000	1697	894.056	178	0,408	259	0,592	437	546	2,25	581	752.706
1.500	1694	895.451	200	0,457	238	0,543	438	538	2,23	560	782.808
2.000	1689	899.860	218	0,493	224	0,507	442	550	2,24	568	777.959
2.500	1682	900.594	230	0,521	211	0,479	441	496	2,12	546	807.342
3.000	1675	906.217	242	0,544	203	0,456	445	495	2,11	543	820.080
3.500	1665	908.821	253	0,563	196	0,437	449	507	2,13	517	867.812
4.000	1655	911.993	261	0,580	189	0,420	450	500	2,11	520	864.727
4.500	1642	913.287	268	0,594	183	0,406	451	514	2,14	510	883.537
5.000	1629	913.543	274	0,606	178	0,394	452	523	2,16	509	887.979
5.500	1614	914.291	279	0,618	173	0,382	452	474	2,05	495	913.765
6.000	1597	914.291	283	0,628	168	0,372	451	456	2,01	493	915.415
6.500	1579	913.335	288	0,637	164	0,363	452	472	2,05	496	910.595
7.000	1560	912.876	291	0,645	160	0,355	451	461	2,02	481	937.275
7.500	1539	909.900	293	0,653	156	0,347	449	462	2,03	480	935.872
8.000	1517	906.838	295	0,660	152	0,340	447	468	2,05	476	938.467
8.500	1494	903.230	296	0,667	148	0,333	444	482	2,09	468	948.554
9.000	1469	895.196	295	0,673	143	0,327	438	426	1,97	451	970.099
9.500	1442	889.540	294	0,679	139	0,321	433	433	2,00	451	960.458
10.000	1414	885.392	293	0,685	135	0,315	428	430	2,00	439	975.428

Tabelle 2.6: einige wichtige Richtgrößen für die Benutzung von verschiedenen Gewichtungen zwischen *perfektem* Hashing und dem Hashing aus Kapitel 2.6.3

können. Dies bedeutet ferner, dass für rund die Hälfte der Tokens nur noch ein Zugriff notwenig ist. Während ohne *Array-Hash* die Anzahl der durchschnittlichen Zugriffe bei 3,61 lag, werden bei dieser Konfiguration nur noch rund 2,12 Zugriffe bezogen auf das gesamte Hashsystem pro *Kookkurrenz-Token* benötigt. Die Anzahl der Zugriffe läßt sich auf etwa 2,0 senken, wenn im *Array-Hash* die Kookkurrenzen für die mindestens 5.500 häufigsten Wörter gezählt werden. Die gesenkte Anzahl der Zugriffe spiegelt sich auch in der benötigten Zeit und der Anzahl der Kollisionen wider (siehe Tabelle 2.6). Während ohne *Array-Hash* für die rund 900.000 Sätze bei etwa 570 Millionen Kollisionen etwa 12 Minuten benötigt werden, kann etwa die gleiche Textmenge, bei eingeschaltetem *Array-Hash*, der Kookkurrenzen zwischen den 9.000 frequentesten Wörtern zählt, mit etwa 426 Millionen Kollisionen in nur etwa 7,5 Minuten verarbeitet werden. Das Einschalten des *Array-Hashes* hat also keine großen Auswirkungen auf die Anzahl der verarbeitbaren Sätze, ohne auslagern zu müssen. Dafür hat er einen enormen Einfluss darauf, wie schnell ein Korpus verarbeitet werden kann.

In diesem Sinne sei noch auf die letzte Spalte der Tabelle 2.6 verwiesen. Während in den Tests zu Tabelle 2.5 auf die Anzahl der verarbeiteten Sätze pro Sekunde geachtet worden ist, stand in der letzten Spalte der Tabelle 2.6 der Fokus darauf, wie viele *Kookkurrenz-Tokens* pro Sekunde verarbeitet werden können. Diese Tabelle kann als ein Richtwert für Zeitabschätzungen anderer Korpora benutzt werden. Dazu muss erst einmal die durchschnittliche Satzlänge n des Korpus berechnet werden. Danach wird die durchschnittliche Anzahl der *Kookkurrenz-Tokens* pro Satz bestimmt. Bei Satzkookkurrenzen bedeutet das beispielsweise, dass jedes Wort eines Satzes mit jedem anderen Wort des Satzes zusammen vorkommt. Formal beschrieben heißt dies $n * (n - 1)$. Unter Ausnutzung der Symmetrie, die eingangs für die Satzkookkurrenzen erklärt worden ist, ergeben sich im Durchschnitt also etwa $\frac{n*(n-1)}{2}$ *Kookkurrenz-Tokens* pro Satz. Bei einer angenommenen Satzlänge von 20 können für jeden Satz $\frac{20*(20-1)}{2} = \frac{380}{2} = 190$ Tokens erwartet werden. Bei einer Konfiguration des *Array-Hashes*, so dass die Kookkurrenzen für die 9000 häufigsten Wörter im *Array-Hash* gezählt werden, können rund 930.000 Kookkurrenzen/Sekunde verarbeitet werden. Ferner bedeutet dies, dass für Satzkookkurrenzen bei einer durschnittlichen Satzlänge von $n = 20$ etwa $\frac{930.000 \text{ Kookkurrenzen/Sekunde}}{190 \text{ Kookkurenzen/Satz}} = 4.900$ Sätze/Sekunde verarbeitet werden können. Diese Spalte dient dazu, dass je nach Satzlänge und benutztem Filter der zu erwartende Durchsatz in Sätzen/Sekunde berechnet werden kann.

2.8 Zusammenfassung

Es konnten in diesem Kapitel zweierlei Erkenntnisse gewonnen werden. Einerseits ist es sinnvoll, *perfektes* und *offenes* Hashen kombiniert einzusetzen. Dadurch können zum Einen mehr als 110 Millionen Kookkurrenzen im Speicher zeitgleich gezählt werden und zum Anderen kann die Anzahl der Kollisionen deutlich reduziert werden. Dies geht einher mit deutlich kürzeren Verarbeitungszeiten. Das Resultat der Optimierungen ist, dass Kookkurrenzen mit nur noch 2 Zugriffen auf den Hash mit über 120 Millionen möglichen Speicherplätzen für Kookkurrenzen gezählt werden können.

Eine weitere Optimierung wird in [ZHW01] beschrieben. Zobel et al haben Wortfrequenzen für einen Korpus mittels eines Hashes berechnet. Dabei haben sie unter anderem die Möglichkeit evaluiert, dass häufige Wörter ggf. so umkopiert werden, dass sie in genau einem Zugriff erreichbar sind. Eine solche *move-to-front*-Strategie hatte bei Zobel's Tests die Hashes zwischen 5% und 40% schneller gemacht. Da bei diesen Tests Wortfrequenzen berechnet worden sind und somit nach dem *Zipfschen Gesetz* etwa die Hälfte aller Wörter mehr als zweimal vorkommen, entsteht vergleichsweise viel Kopieraufwand. Für Kookkurrenzen besagt das *Zipfsche Gesetz*, dass nur etwa $\frac{1}{4}$ aller Kookkurrenzen mehr als einmal vorkommen 1.4. Dementsprechend wird der Kopieraufwand deutlich geringer sein. Durch diese *move-to-frond*-Strategie sollten durchschnittliche Zugriffe von deutlich unter 2 möglich sein.

Eine weitere Optimierung besteht aus der Beobachtung, dass bei eingeschaltetem *Array-Hash* im *RAM-Hash* kaum Kookkurrenzen gezählt werden, die häufiger als 255mal beobachtet werden können. Bei den Tests aus diesem Kapitel wurden im *RAM-Hash* immer $s_{RAMHash,Frequenz} = 2$ benutzt. Damit können Frequenzen bis zu $2^{2*8} = 2^{16} = 65535$ gezählt werden. Die Frage ist also: Wie groß muß der *Array-Hash* gewählt werden, so dass im *RAM-Hash* nur noch Frequenzen von nicht mehr als 255 gezählt werden? Dann könnte $s_{RAMHash,Frequenz}$ auf eins gesetzt werden. Der *RAM-Hash* würde dadurch rund 100 MB kompakter werden. Der zusätzliche Speicher würde dann wiederum Platz für rund 14 Millionen neue Hashwerte liefern. Aus den bisherigen Erfahrungen mit *Medusa* würde das bedeuten, dass zwischen 100.000 und 200.000 Sätze mehr verarbeitet werden können, ohne dass der *RAM-Hash* auf die Festplatte ausgelagert werden muss. Ferner ist zu erwarten, dass für die weitere Verarbeitung der Kookkurrenzen durch *Medusa* die Festplatten-IO, um etwa 15% reduziert werden kann.

Kapitel 3

Signifikanzmaße

In Kapitel 1.1 wurde definiert, was eine *Kookkurrenz* ist, Kapitel 1.2 erklärt, warum das Messen von Kookkurrenzen sinnvoll ist und in Kapitel 2 wurde gezeigt, wie Kookkurrenzen auch auf größeren Korpora effektiv berechnet werden können. In diesem Kapitel werden Ansätze vorgestellt, die darüber entscheiden, welche der gefundenen *Kookkurrenzen* als signifikant markiert und damit für eine weitere Verarbeitung genutzt werden sollen.

Auf das Beispiel aus Kapitel 2 bezogen, welches eingangs vorgestellt wurde, bedeutet dies, dass in rund 11,35 Millionen Sätzen etwa 1,47 Mrd. *Kookkurrenzen* gefunden werden. Die meisten dieser *Kookkurrenzen* sind jedoch willkürlich auftretende Artefakte, die nichts miteinander zu tun haben (vgl. Kapitel 1.2). Denn es ist möglich, zu jedem beliebigen Paar zweier miteinander unverwandter Wörter einen sinnvollen Satz zu formulieren, in welchem beide vorkommen. Dementsprechend können aus den 1,47 Mrd. *gefundenen Kookkurrenzen* etwa 30 Millionen *signifikante Kookkurrenzen* bestimmt werden.

Eine Methode, welcher darüber entscheidet, ob eine *Kookkurrenz* wichtig oder unwichtig ist, wird *Signifikanzmaß* genannt. Um darüber zu entscheiden, ob eine *Kookurrenz* wichtig ist, können je nach *Signifikanzmaß* statistische Größen wie die *Frequenz der Kookkurrenz*, die *Frequenzen der involvierten Wörter* und der *Größe des Korpus* eine Rolle spielen.

Im Folgenden sollen ausschließlich *Kookkurrenzen zwischen zwei Wörtern* betrachtet werden. Die beiden Wörter dieser *Kookkurrenz* werden mit a und b bezeichnet. Durch n_a und n_b werden deren *Wortfrequenzen* beschrieben. Das gemeinsame Auftreten (*Frequenz der Kookkurrenz*) wird im Folgenden mit n_{ab} bezeichnet. Die Größe des Korpus wird durch n repräsentiert.

Für die Berechnung von signifikanten Kookkurrenzen wurden die ersten fünf Millionen Sätze des Korpus *de* benutzt. Aus diesen fünf Millionen Sätzen konnten insgesamt rund 350 Millionen verschiedene Kookkurrenzen ex-

trahiert werden. 76,5% dieser Kookkurrenzen traten nur einmal auf (vgl. *Zipfsches Gesetz* für *Kookkurrenzen* in Kapitel 1.4).

Im Folgenden werden aus den rund 350 Millionen Kookkurrenzen die eine Million signifikantesten und die ein bzw. zehn Millionen unsignifikantesten Kookkurrenzen grafisch dargestellt. Sie stellen damit lediglich den Anfang bzw. das Ende einer signifikanzsortierten Kookkurrenzliste dar.

3.1 Ausgewählte Signifikanzmaße

Eine gute Übersicht über *Signifikanzmaße* wird in [Pec05] gegeben. Darin listet Pavel Pecina 57 verschiedene *Signifikanzmaße* auf. Die Wichtigsten werden in diesem Kapitel vorgestellt.

Das einfachste aller Signifikanzmaße ist eine *frequenzsortierte Liste*. Wie in Formel 3.1 zu sehen ist, spielt dabei ausschließlich die *Frequenz des gemeinsamen Auftretens* eine Rolle. Durch einen Schwellwert s_{freq} werden diejenigen *Kookkurrenz* herausgefiltert, welche den Schwellwert unterschreiten. Aus dem *Zipfschen Gesetz für Kookkurrenzen* kann abgeleitet werden, dass bei einem Schwellwert $s_{freq} \geq 2$ bereits 75% aller Kookkurrenzen herausgefiltert werden.

$$sig_{freq}(n_{ab}) = n_{ab} \tag{3.1}$$

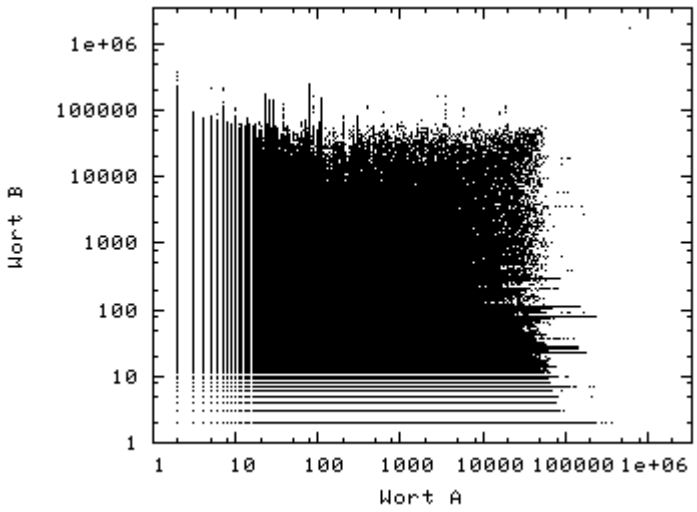

Abbildung 3.1: die eine Million *signifikantesten Kookkurrenzen* nach sig_{freq}

Abbildung 3.1 zeigt die eine Million *signifikantesten Kookkurrenzen* nach sig_{freq} aus dem eingangs definierten fünf Millionen Satz-Korpus. Auf den Achsen sind die beiden Wörter abgetragen, die jeweils frequenzsortiert sind (vgl. *Zipfsches Gesetz* aus Kapitel 1.4). Abbildung 3.1 zeigt, dass die signifikantesten Kookkurrenzen nach sig_{freq} diejenigen sind, welche auf sehr frequenten Wörtern basieren.

Die unsignifikantesten Kookkurrenzen nach sig_{freq} sind diejenigen, welche einmal vorkommen. Für den Fünf-Millionen-Satz-Korpus sind dies rund 272 Millionen. Aus dieser Menge der unsignifikantesten Kookkurrenzen wurden zufällig zehn Millionen ausgewählt. Abbildung 3.2 zeigt die nach sig_{freq} unsignifikantesten Kookkurrenzen, welche zwischen nieder- und mittelfrequenten Wörtern aufgetreten sind.

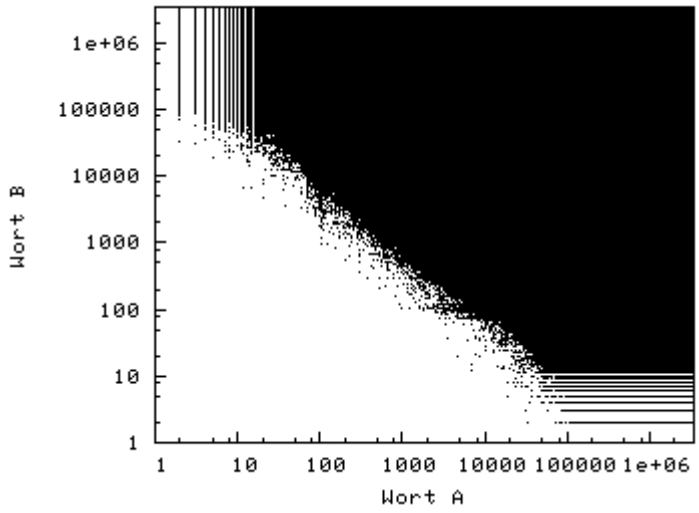

Abbildung 3.2: zehn Millionen *unsignifikante Kookkurrenzen* mit $sig_{freq} = 1$ (Auswahl aus 272 Millionen Kookkurrenzen)

Kritik an diesem Maß besteht darin, dass es keine Abhängigkeiten zu n_a, n_b oder n_{ab} gibt. Treten die beiden Wörter einer Kookkurrenz hinreichend oft auf, beispielsweise $n_a = n_b = 10000$, so ist zu erwarten, dass beide Wörter auch oft miteinander auftreten können (also beispielsweise 90mal). Hingegen treten beide Wörter nur $n_a = n_b = 10$mal auf und davon siebenmal gemeinsam, so ist dies wesentlich auffälliger. Nach dem Signifikanzmaß sig_{freq} wird aber die erste Kookkurrenz als signifikanter erachtet.

3.1.1 Dice

Anders als das Signifikanzmaß der *frequenzsortierten Liste* sig_{freq} benutzt der *Dice-Koeffizient* sig_{dice} für das Berechnen der *Signifikanz* neben der *Frequenz des gemeinsamen Auftretens* n_{ab} auch die *Frequenzen der beiden Wörter* n_a und n_b (siehe Formel 3.2 - vgl. [Dic45] und [Lan04]).

$$sig_{dice}(n_a, n_b, n_{ab}) = \frac{2 * n_{ab}}{n_a + n_b} \tag{3.2}$$

Der *Dice-Koeffizient* läßt sich aus dem *Harmonischen Mittel* ableiten (vgl. [Eve04]). Bei näherer Betrachtung des *Dice-Koeffizienten* aus Formel 3.2 zeigt sich, dass eine Signifikanz berechnet wird, die im Bereich von $[0, 1]$ liegt.

In Abbildung 3.3 sind die eine Million *unsignifikantesten Kookkurrenzen* dargestellt, die mit dem *Dice*-Koeffizient aus Formel 3.2 berechnet worden sind. Die Abbildung zeigt, für die eine Million unsignifikantesten Kookkurrenzen nach dem *Dice*-Koeffizienten aus dem Bereich $0 \leq sig_{dice} \leq 10^{-6}$, dass sie zwischen Wörter auftreten, die einerseits den Rang 1 und andererseits einen Rang größer als etwa 10.000 haben.

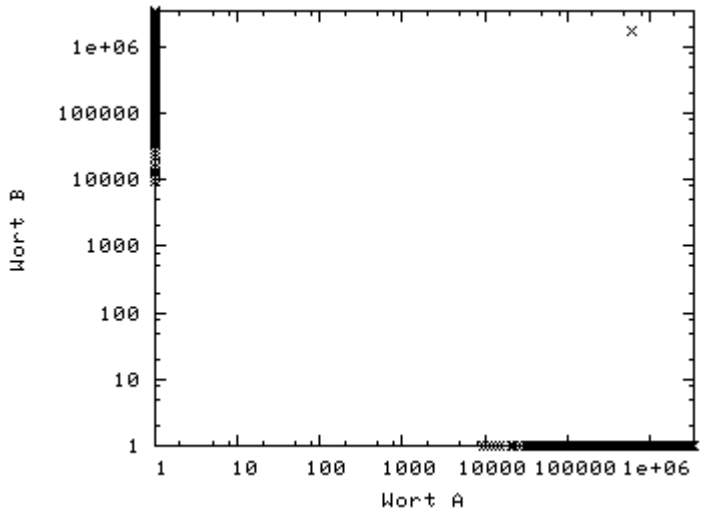

Abbildung 3.3: die eine Million *unsignifikantesten Kookkurrenzen* nach sig_{dice}

Werden anstelle der einen Million die zehn Millionen unsignifikantesten Kookkurrenzen betrachtet (siehe Abbildung 3.4), dann treten diese zwischen einem Wort vom Rang kleiner als 12 und einem zweiten Wort vom Rang größer 10.000 auf.

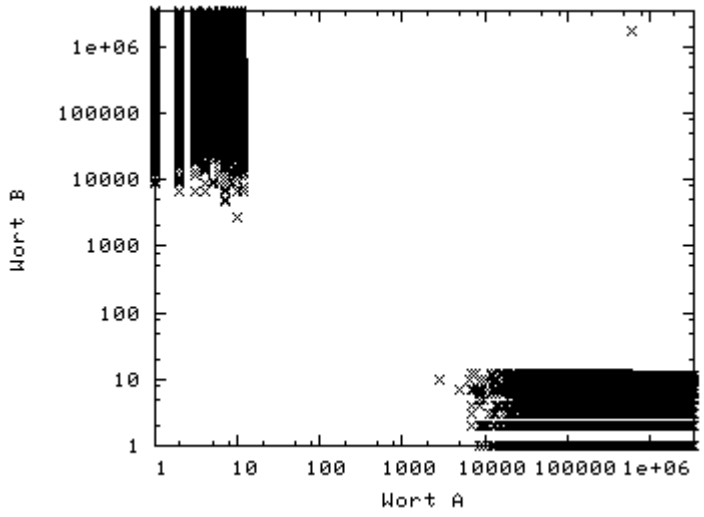

Abbildung 3.4: die zehn Millionen *unsignifikantesten Kookkurrenzen* nach sig_{dice}

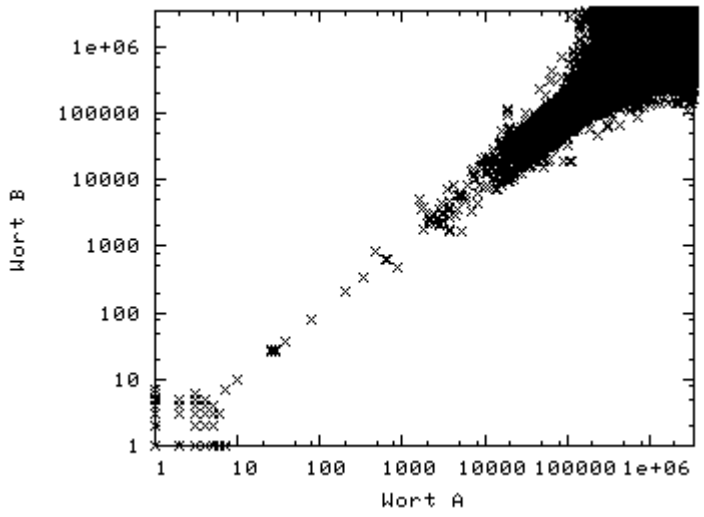

Abbildung 3.5: die eine Million *signifikantesten Kookkurrenzen* nach sig_{dice}

Abbildung 3.5 zeigt die eine Millionen signifikantesten Kookkurrenzen, die vorrangig zwischen niederfrequenten Wörtern auftreten. Weiterhin zeigt die Abbildung, dass für die meisten signifikanten Kookkurrenzen, die häufiger als einmal aufgetreten sind, $n_a = n_b$ gilt.

Die wichtigste Eigenschaft des *Dice*-Koeffizienten ist, dass Kookkurrenzen als unsignifikant bewertet werden, wenn sie einmal bzw. selten zusammen aufgetreten sind und eines der beiden Wörter *hochfrequent* und das zweite Wort *niederfrequent* ist. Auch wenn an dieser Stelle die Eigenschaft der Unsignifikanzbewertung des *Dice*-Koeffizienten nicht wichtig erscheint, so wird in der Zusammenfassung auf sie zurückgegriffen.

3.1.2 Jaccard

Genau wie der *Dice-Koeffizient* $sig_{dice}(n_a, n_b, n_{ab})$ aus Kapitel 3.1.1 benutzt auch der *Jaccard-Koeffizient* neben der *Frequenz des gemeinsamen Auftretens* n_{ab} die *Frequenzen der beiden Wörter* n_a und n_b zur Berechnung der Signifikanz (siehe Formel 3.3 - vgl. [Tan58] und [Lan04]).

$$sig_{jaccard}(n_a, n_b, n_{ab}) = \frac{n_{ab}}{n_a + n_b - n_{ab}} \tag{3.3}$$

In Evert (vgl. [Eve04] Seite 56) wird gezeigt, dass es zwischen dem *Jaccard-Koeffizient* und dem *Dice-Koeffizient* eine bijektive Abbildung der Form

$$sig_{jaccard}(n_a, n_b, n_{ab}) = \frac{sig_{dice}(n_a, n_b, n_{ab})}{2 - sig_{dice}(n_a, n_b, n_{ab})} \tag{3.4}$$

gibt. Abbildung 3.6 zeigt, dass es zwar keine lineare Abhängigkeit zwischen dem *Dice*- und dem *Jaccard*-Koeffizienten gibt, aber die Abbildung macht deutlich, dass sich beide Koeffizienten proportional zueinander verhalten.

$sig_{jaccard}$ liefert dementsprechend auch die gleichen signifikanten *Kookkurrenzen*. Sie unterscheiden sich nur im Signifikanzwert, der entsprechend höher oder niedriger ist. Die relative Ordnung der Kookkurrenzen, bezogen auf ihren Signifikanzwert, bleibt dabei komplett erhalten. Daher liefert der *Jaccard*-Koeffizient auch die gleichen Plots wie der *Dice*-Koeffizient in den Abbildungen 3.3 bis 3.5.

3.1.3 Piatersky-Shapiro-Maß

Während die Signifikanzmaße sig_{dice} und $sig_{jaccaard}$ sich aus dem *harmonischen Mittel* ableiten lassen, steht beim Maß von *Piatersky und Shapiro* die stochastische Unabhängigkeit aus Formel 3.5 im Mittelpunkt.

$$P(n_{ab}) = P(n_a) * P(n_b) \tag{3.5}$$

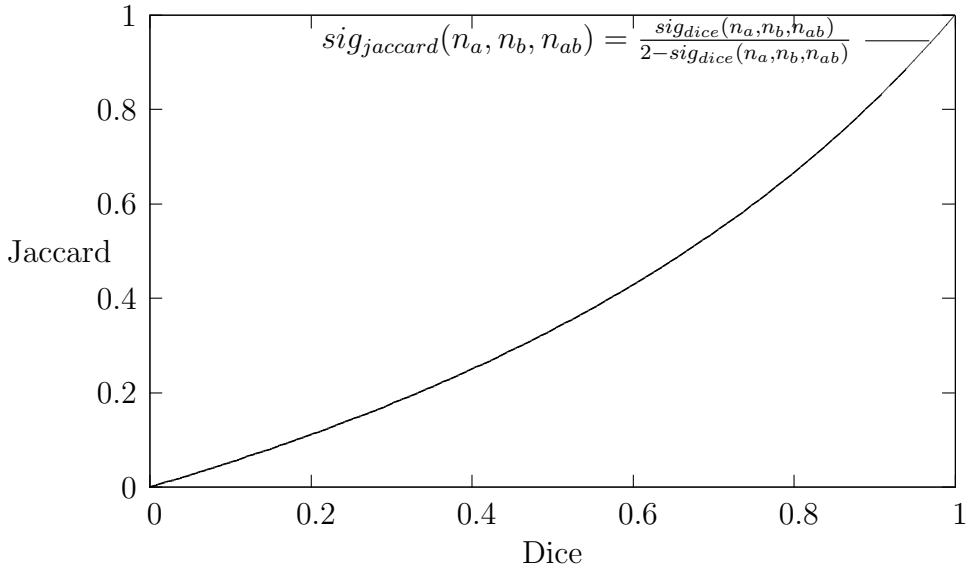

Abbildung 3.6: Abhängigkeit zwischen dem *Dice*- und dem *Jaccard*-Koeffizienten

Dabei entspricht $P(n_a) * P(n_b)$ der *erwarteten Wahrscheinlichkeit* für das gemeinsame Auftreten der Wörter a und b. $P(n_{ab})$ steht für die real *beobachtete Wahrscheinlichkeit*. Gilt die Gleichung aus Formel 3.5, dann wird von *stochastischer Unabhängigkeit* gesprochen. Dabei entspricht die *beobachtete Wahrscheinlichkeit* für das gemeinsame Auftreten von a und b der *erwarteten Wahrscheinlichkeit*. In diesem Fall sind a und b zufällig aufgetreten und somit nicht *signifikant*. Je größer die Differenz zwischen $P(n_{ab})$ und $P(n_a) * P(n_b)$ ist, um so signifikanter ist das gemeinsame Auftreten der beiden Wörter. Aus dieser Überlegung definiert sich auch das Maß von *Piatersky und Shapiro* aus Formel 3.6.

$$sig_{Piatersky}(n_a, n_b, n_{ab}, n) = P(n_{ab}) - P(n_a) * P(n_b) = \frac{n_{ab}}{M} - \frac{n_a * n_b}{N^2} \quad (3.6)$$

M entspricht dabei der Summe aller Kookkurrenzfrequenzen und N der Summe aller Wortfrequenzen. N lässt sich aus der *Anzahl der Sätze* n und der *durchschnittlichen Satzlänge* l_{Satz} durch $N = n * l_{Satz}$ approximieren. Auch M lässt sich aus den beiden Größen n und l_{Satz} approximieren. Aus einem durchschnittlichen Satz der Länge l_{Satz} können $l_{Satz} * (l_{Satz} - 1)$ Kookkurrenzen extrahiert werden. M kann für Satzkookkurrenzen dementsprechend mit $M = n * l_{Satz} * (l_{Satz} - 1)$ approximiert werden. Dies in Formel 3.6 eingesetzt, liefert das vereinfachte Maß für Satzkookkurrenzen aus Formel 3.7.

$$
\begin{aligned}
sig_{Piatersky}(n_a, n_b, n_{ab}, n) &= \frac{n_{ab}}{n * l_{Satz} * (l_{Satz} - 1)} - \frac{n_a * n_b}{(n * l_{Satz})^2} \\
&= n_{ab} - \frac{n_a * n_b * n * l_{Satz} * (l_{Satz} - 1)}{n^2 * l_{Satz}^2} \\
&= n_{ab} - \frac{n_a * n_b}{n} * \frac{l_{Satz} * (l_{Satz} - 1)}{l_{Satz}^2} \\
&\approx n_{ab} - \frac{n_a * n_b}{n}
\end{aligned}
\tag{3.7}
$$

$\frac{n_a * n_b}{n}$ steht dabei für die '*erwartete Frequenz*' des gemeinsamen Auftretens der beiden Wörter. Für $n = 5.000.000$ Sätze wird dieser Quotient für mittel- und niederfrequente Wörter immer kleiner als 1 sein, so dass selbst das einmalige gemeinsame Auftreten zweier Wörter als signifikant bewertet wird. Für den Fall $n_a = n_b$ müssen beide Wörter eine Frequenz von

$$\sqrt{5000000} = 2.237^1$$

haben, damit eine beobachtete Frequenz von 1 unter der erwarteten Frequenz bleibt. Dieses Problem der Signifikanz seltener Wörter wird von den Maßen ab dem Kapitel 3.1.4 zu lösen versucht.

3.1.4 Mutual Information

Genau wie das Maß von *Piatersky und Shapiro* basiert auch die *Mutual Information* auf der stochastischen Unabhängigkeit. Die Berechnung der Signifikanz nach $sig_{Piatersky}$ gestaltet sich aus numerischer Sicht bei größeren Korpora als schwierig, da die Wahrscheinlichkeiten sehr klein werden. Dieses Problem löst die *Mutual Information*, indem von beiden Seiten aus Formel 3.5 nicht mit der Wahrscheinlichkeit selber, sondern mit deren Logarithmus gerechnet wird (vgl. Formel 3.8).

$$\log_2 P(n_{ab}) = \log_2 P(n_a) * P(n_b) \tag{3.8}$$

Genau wie beim Maß von *Piatersky und Shapiro* interessiert auch bei der *Mutual Information* die Differenz bzw. der Quotient zwischen *beobachteter* und *erwarteter* Wahrscheinlichkeit. Aus der Formel 3.8 kann die *Mutal Information* schließlich durch Umstellen wie in Formel 3.9 gewonnen werden.

[1]Bezogen auf den Fünf-Millionen-Satz-Korpus bedeutet eine Frequenz von 2.237 einen Rang von etwa 2.800 bei ungefähr 3,8 Millionen Wörtern.

$$
\begin{aligned}
sig_{MI}(n_a, n_b, n_{ab}, n) &= \log_2 P(n_{ab}) - \log_2 P(n_a) * P(n_b) \\
&= \log_2 \frac{P(n_{ab})}{P(n_a) * P(n_b)}
\end{aligned} \tag{3.9}
$$

Eine andere Ableitung der *Mutual Information* kommt aus der *Informationstheorie* (vgl. Formel 3.10). Die *Mutual Information* entspricht dabei der Differenz der Entropie $H(n_a)$ und der bedingten Entropie $H(n_a|n_b)$. Die *Mutual Information* kann somit als der *Informationsgewinn* aufgefasst werden.

$$
sig_{MI}(n_a, n_b, n_{ab}, n) = H(n_a) - H(n_a|n_b) = H(n_b) - H(n_b|n_a) \tag{3.10}
$$

Eine *Mutual Information* von $sig_{MI} = 3$ bedeutet, dass die beiden Wörter a und b $2^3 = 8$mal häufiger beobachtet worden sind als es aufgrund der Wortfrequenzen zu erwarten war. Church (vgl. [CH90]) meint, dass eine *Mutual Information* von mindestens drei gefordert werden muss, um signifikante Kookkurrenzen zu bestimmen. Für den Fall, dass zwei Wörter, die einmal vorgekommen sind und auch innerhalb des gleichen Fensters beobachtet werden, gilt für die *Mutual Information* approximativ $sig_{MI} = \log_2 n$. Auf den Fünf-Millionen-Satz-Korpus bezogen, bedeutet dies, dass eine *Mutual Information* von $sig_{MI} = \log_2 5.000.000 \approx 22$ berechnet wird, d. h. das gemeinsame Auftreten der beiden seltenen Wörter ist $2^{22} = 4.194.304$-mal wahrscheinlicher, als erwartet werden konnte. Aus den gleichen Gründen wie beim Maß von *Piatersky und Shapiro* werden durch die Annahme der *stochastischen Unabhängigkeit* seltene Ereignisse bevorzugt.

Abbildung 3.7 zeigt die eine Million signifikantesten Kookkurrenzen nach sig_{MI}. Aus diesem Plot ist ersichtlich, dass die signifikantesten Kookkurrenzen zwischen einmal vorkommenden Wörtern beobachtet werden können.

Die Abbildungen 3.8 und 3.9 zeigen die eine Millionen bzw. zehn Millionen unsignifikantesten Kookkurrenzen nach sig_{MI}. Es kann diesen Plots entnommen werden, dass es egal ist, ob die eine oder zehn Millionen unsignifikantesten Kookkurrenzen betrachtet werden, da sich die Plots kaum geändert haben. Interessant an beiden Grafiken ist jedoch, dass im Bereich der Wortnummer von 1-100 weitestgehend keine Kookkurrenzen als unsignifikant berechnet werden. Dementsprechend benachteiligt die *Mutual Information* nicht hochfrequente Kookkurrenzen.

Bezogen auf die Signifikanz jedoch sind aufgrund des Verhältnisses von *beobachtetem* und *erwartetem* gemeinsamen Auftreten seltenere Ereignisse bevorzugt. Church (vgl. [CH90]) meint, dass die Wörter nicht beobachtet werden sollten, die seltener als fünfmal vorgekommen sind. Dies ist jedoch

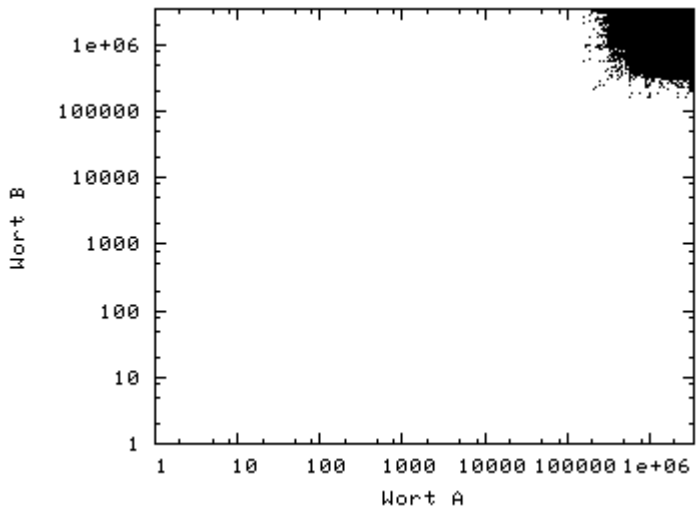

Abbildung 3.7: die eine Million *signifikantesten Kookkurrenzen* nach sig_{MI}

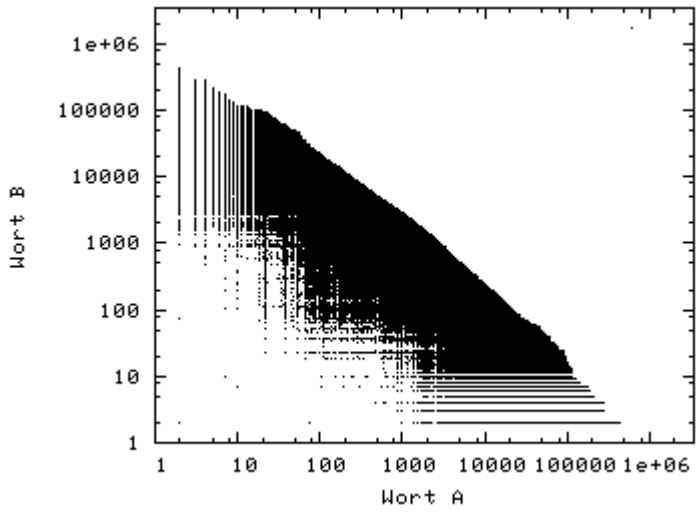

Abbildung 3.8: die eine Million *unsignifikantesten Kookkurrenzen* nach sig_{MI}

bedenklich, da nach dem *Zipfschen Gesetz* rund 83% aller Wörter ignoriert werden würden.

Eine Lösung für das Problem der *Mutual Information*, dass die signifikantesten Kookkurrenzen sehr seltene Ereignisse sind, liefert die Kombination der beiden Maße sig_{MI} und $sig_{Frequenz}$. Die Abbildung 3.1 zeigt, dass

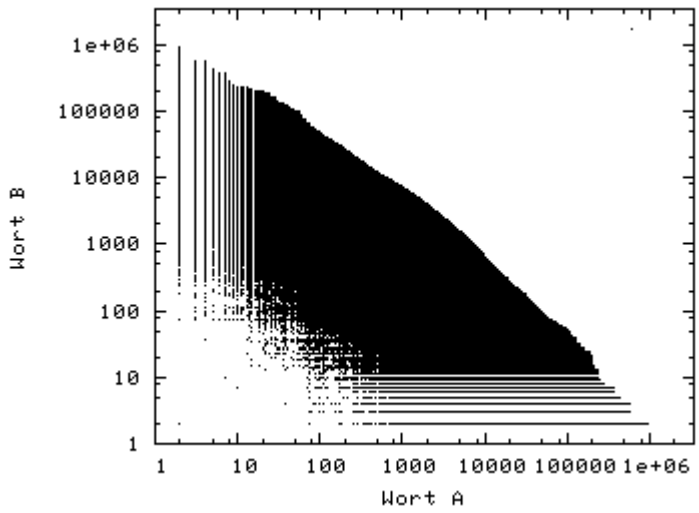

Abbildung 3.9: die zehn Millionen *unsignifikantesten Kookkurrenzen* nach sig_{MI}

$sig_{Frequenz}$ hochfrequente Kookkurrenzen bevorzugt, während Abbildung 3.7 zeigt, dass die *Mutual Information* die niederfrequenten Kookkurrenzen bevorzugt. Das Produkt beider Maße wie in Formel 3.11 wird *Local Mutual Information* genannt (vgl. [Eve04]).

$$\begin{aligned} sig_{LMI}(n_a, n_b, n_{ab}, n) &= sig_{Frequenz}(n_{ab}) * sig_{MI}(n_a, n_b, n_{ab}, n) \\ &= n_{ab} * \log_2 \frac{P(n_{ab})}{P(n_a) * P(n_b)} \\ &= n_{ab} * \log_2 \frac{n_{ab} * n}{n_a * n_b} \end{aligned} \tag{3.11}$$

Auf die *Local Mutual Information* wird im folgenden Kapitel näher eingegangen. Es sei jedoch auf die Ähnlichkeit zur Termgewichtung $tf * IDF$ aus dem *Information Retrieval* (vgl. [Sal89]) hingewiesen. Da die IDF (Inverse Document Frequency) für Kookkurrenzen nicht sinnvoll ist, ist die Ersetzung dieser durch die *Mutual Information* ein guter Ersatz.

3.1.5 Poisson-Maß

Ein weiterer Ansatz zur Berechnung von signifikanten Kookkurrenzen basiert auf der Poisson-Verteilung aus Formel 3.12.

$$P(n, k) = \frac{1}{k!} \lambda^k e^{-\lambda} \tag{3.12}$$

In Anlehnung an die stochastische Unabhängigkeit aus Formel 3.5 berechnet sich $\lambda = \frac{n_a * n_b}{n}$. Dabei kann λ als die erwartete Frequenz n_{ab} aufgefaßt werden. Die Poisson-Verteilung scheint an dieser Stelle für das Berechnen von Signifikanzen recht gut geeignet, da sie auch die „Verteilung der seltenen Ereignisse“ genannt wird und dementsprechend erwartet werden kann, dass sie mit den rund 75% der Kookkurrenzen besser umgehen kann, die nur einmal beobachtet werden konnten.

Quasthoff und Wolff (vgl. [QW02]) schlagen daher auf Basis der Poisson-Verteilung das *Poisson*-Maß aus Formel 3.13 vor.

$$sig_{Poisson}(n_a, n_b, k, n) = \frac{k * (\log k - \log \lambda - 1)}{\log n} \tag{3.13}$$

Wobei n wieder für die Anzahl der Sätze und k für die Anzahl des gemeinsamen Auftretens der beiden Wörter (bisher immer mit n_{ab} beschrieben) steht.

Abbildung 3.10 zeigt die eine Million signifikantesten Kookkurrenzen, die durch das Poisson-Maß $sig_{Poisson}$ aus dem eingangs definierten Fünf-Millionen-Satz-Korpus berechnet werden konnten. Bei einem Vergleich dieser Abbildung mit der Abbildung 3.1 der signifikantesten Kookkurrenzen nach sig_{Freq} ist erkennbar, dass beide Darstellungen sich sehr ähnlich sind.

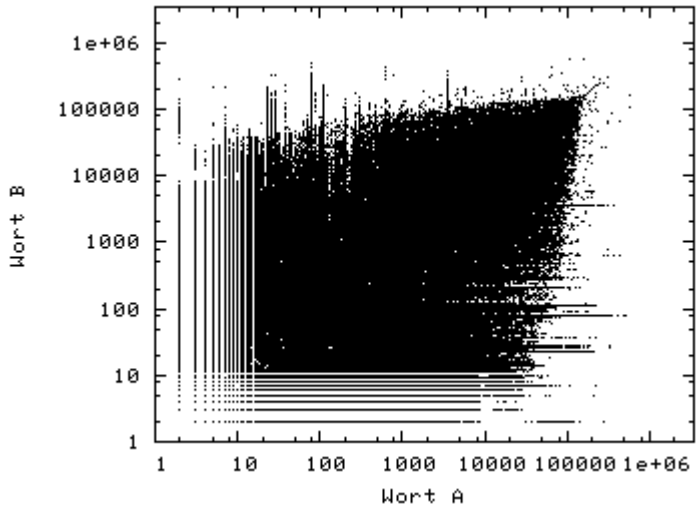

Abbildung 3.10: die eine Million *signifikantesten Kookkurrenzen* nach $sig_{Poisson}$

Die beiden Abbildungen 3.11 und 3.12 zeigen die eine bzw. zehn Millionen unsignifikantesten Kookkurrenzen. Auch diese beiden Plots zeigen be-

reits Bekanntes. Ein Vergleich der beiden Plots 3.8 und 3.9, welche die unsignifikantesten Kookkurrenzen der *Mutual Information* abbilden, zeigt, dass das Poisson-Maß und die *Mutual Information* sehr ähnliche *unsignifikante Kookkurrenzen* berechnen. An dieser Stelle muss jedoch darauf hingewiesen werden, dass in beiden Fällen als *unsignifikante Kookkurrenzen*, diejenigen berechnet werden, bei denen die beobachtete Frequenz am deutlichsten unter der erwarteten Frequenz bleibt. Es wird bei diesen Kookkurrenzen deshalb auch von signifikant seltenen Ereignissen gesprochen.

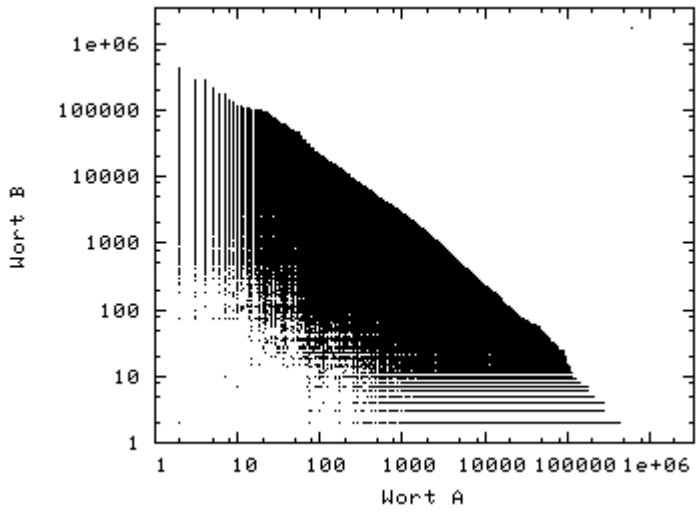

Abbildung 3.11: die eine Million *unsignifikantesten Kookkurrenzen* nach $sig_{Poisson}$

Es konnte anhand der Plots gezeigt werden, dass das Poisson-Maß bei den signifikanten Kookkurrenzen dem Maß $sig_{Frequenz}$ und bei den unsignifikanten Kookkurrenzen der Mutual Information sig_{MI} ähnlich ist. Dies läßt sich auch im Poisson-Maß bei genauerem Hinschauen zeigen. Aus der Formel 3.13 wird zuerst $\log k - \log \lambda$ zusammengefaßt. Das Ergebnis ist die Formel 3.14.

$$sig_{Poisson}(n_a, n_b, k, n) = \frac{k * (\log \frac{k}{\lambda} - 1)}{\log n} \tag{3.14}$$

Zu Beginn wurde λ als $\lambda = \frac{n_a * n_b}{n}$ definiert. Wird in Formel 3.14 λ entsprechend ersetzt, dann kann daraus die Formel 3.15 abgeleitet werden. Weiterhin wird k wieder durch die eingangs definierte Notation n_{ab} ersetzt. Dies ist insofern unbedenklich, da beide Bezeichner das Gleiche beschreiben. Das Ergebnis aus dieser Umstellung ist die Formel 3.15.

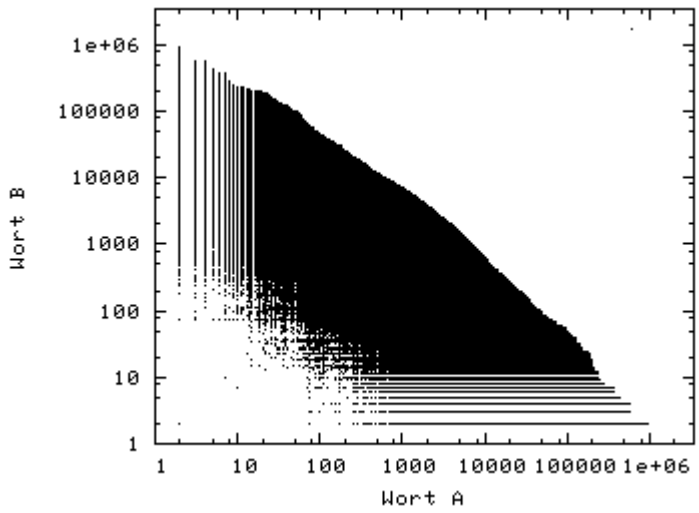

Abbildung 3.12: die zehn Millionen *unsignifikantesten Kookkurrenzen* nach $sig_{Poisson}$

$$\begin{aligned} sig_{Poisson}(n_a, n_b, n_{ab}, n) &= \frac{n_{ab} * (\log \frac{n_{ab}}{\frac{n_a * n_b}{n}} - 1)}{\log n} \\ &= \frac{n_{ab} * (\log \frac{n_{ab} * n}{n_a * n_b} - 1)}{\log n} \end{aligned} \tag{3.15}$$

In der letzten Zeile der Formel 3.15 ist schließlich der oben beschriebene Zusammenhang ersichtlich. Dies wird durch eine letzte Umformung zu Formel 3.16 noch deutlicher.

$$\begin{aligned} sig_{Poisson}(n_a, n_b, n_{ab}, n) &= \frac{n_{ab} * (\log \frac{n_{ab} * n}{n_a * n_b} - 1)}{\log n} \\ &= \frac{n_{ab} * \log \frac{n_{ab} * n}{n_a * n_b} - n_{ab}}{\log n} \\ &= \frac{sig_{LMI}(n_a, n_b, n_{ab}, n) - sig_{Frequenz}(n_{ab})}{\log n} \end{aligned} \tag{3.16}$$

Die Formel 3.16 zeigt, dass das Poisson-Maß letztlich auf die Differenz zwischen sig_{LMI} und $sig_{Frequenz}$ reduziert werden kann. Wobei Beobachtungen die Vermutung erlauben, dass $sig_{Frequenz}$ einen eher kleinen Einfluss hat.

Die oben gemachten Vergleiche der Plots hatten Zusammenhänge zwischen $sig_{Poisson}$ und $sig_{Frequenz}$ sowie $sig_{Poisson}$ und sig_{MI} vermuten lassen. Diese Zusammenhänge finden sich im Maß der *Local Mutual Information* sig_{LMI} wieder, welches ausschließlich auf diese beiden Maße zurückgreift.

Es ist jedoch offensichtlich, dass die *Local Mutual Information* vorrangig von $sig_{Frequenz}$ abhängig ist. Während $sig_{Frequenz}$ auf den Fünf-Millionen-Satz-Korpus bezogen, Werte von ein bis knapp fünf Millionen zulässt, so skaliert die *Mutual Information* aufgrund des Logarithmus deutlich weniger.

Um die Abhängigkeit des Poisson-Maßes und damit der *Local Mutual Information* von der Frequenz einer Kookkurrenz zu bestimmen, kann die Korrelation nach Pearson (vgl. [Boh91]) berechnet werden. Da sowohl die *Local Mutual Information* und damit auch das Poisson-Maß anders mit der Bewertung der Signifikanz von Kookkurrenzen als $sig_{Frequenz}$ umgehen, die deutlich seltener aufgetreten sind als erwartet, wurden zwei Korrelationen berechnet.

Bei der ersten Berechnung wurde die Korrelation zwischen $sig_{Frequenz}$ und $sig_{Poisson}$ berechnet. Dabei wurde eine Korrelation von 0,82 berechnet. Dies zeigt, dass das Poisson-Maß sehr stark von der Frequenz abhängig ist.

In der zweiten Korrelationsberechnung wurden diejenigen Kookkurrenzen nicht für die Berechnung der Korrelation berücksichtigt, die seltener beobachtet worden sind, als es zu erwarten war. Die Motivation dafür liegt darin begründet, dass beide Maße unterschiedlich damit umgehen. Einerseits werden nach dem Poisson-Maß solche Kookkurrenzen mit einem negativen Signifikanzwert berechnet. Andererseits können nach $sig_{Frequenz}$ solche Kookkurrenzen nicht unterschieden werden. Unter diesen Umständen kann sogar eine Korrelation von 0,92 berechnet werden.

3.1.6 Log-Likelihood-Maß

Das letzte Maß, welches in diesem Kapitel vorgestellt werden soll, basiert auf der Binomialverteilung (vgl. [Dun93]) aus Formel 3.17. n steht dabei für die Anzahl der Sätze und k für das gemeinsame Auftreten zweier Wörter. p leitet sich aus der Unabhängigkeitsannahme $p = p(n_a) * p(n_b)$ ab.

$$p(k, n) = \binom{n}{k} p^k (1-p)^{n-k} \tag{3.17}$$

Eine wichtige Einschränkung, die Dunning beim Log-Likelihood-Maß jedoch aufgrund der Binomialverteilung machen muss, ist $k \leq n$, da sonst der Binomialkoeffizient $\binom{n}{k}$ nicht berechnet werden kann. Um dies in jedem Fall sicherstellen zu können, müssten jedoch die Wörter in dem zu beobachtenden

Fenster distinktet werden. Dies wiederum reduziert die Vielfalt einer Sprache. In einem Satz, in welchem die Wendung *viel zu viel* vorkommt, ist diese Wendung jedoch signifikant. In Kapitel 4.2 wird eine Lösung angeboten, die die Bedingung $k \leq n$ in der praktischen Anwendung erfüllt, ohne dabei eine Distinktion[2] der Wörter innerhalb eines Fensters machen zu müssen.

Basierend auf der *Binomialverteilung* berechnet Dunning (vgl. [Dun93]) das Likelihood-Ratio. Dieses Verhältnis sagt genau wie bei der *Mutual Information* etwas darüber aus, wieviel die Beobachtung zweier Wörter von dem abweicht, was erwartet werden konnte.

Aus der Binomialverteilung (siehe Formel 3.17) kann schließlich das Log-Likelihood-Maß aus Formel 3.18 abgeleitet werden (vgl. [Dun93]).

$$sig_{LGL} = 2 \left[\begin{array}{l} n \log n - n_A \log n_A - n_B \log n_B + n_{AB} \log n_{AB} \\ + (n - n_A - n_B + n_{AB}) \log (n - n_A - n_B + n_{AB}) \\ + (n_A - n_{AB}) \log (n_A - n_{AB}) + (n_B - n_{AB}) \log (n_B - n_{AB}) \\ - (n - n_A) \log (n - n_A) - (n - n_B) \log (n - n_B) \end{array} \right] \tag{3.18}$$

Die Abbildung 3.13 zeigt die eine Million unsignifikantesten Kookkurrenzen nach sig_{lgl}. Es ist ersichtlich, dass sich für Kookkurrenzen mit höheren Wortnummern „Linien“ bilden, deren Abstände zueinander sukzessive größer werden.

Auf den ersten Blick scheinen das Log-Likelihood-Maß und das Poisson-Maß ähnlich zu funktionieren. Das liegt darin begründet, dass sich bei einem großen n und $0 \leq p \leq \min\{\frac{10}{n}, \frac{n}{1500}\}$ beide Verteilungen annähern (vgl. [Bar99]). Andererseits zeigt ein Vergleich der beiden Plots 3.11 und 3.13, dass beide Maße bei den eine Million unsignifikantesten Kookkurrenzen offensichtlich geringfügig unterschiedliche Ergebnisse erzielen. Der Grund dafür liegt in der Behandlung von Kookkurrenzen, die seltener beobachtet werden konnten, als es zu erwarten war. In einem Beispiel sind die Wortfrequenzen $n_a = n_b = 100.000$, die Größe des Korpus $n = 5.000.000$ und beide Wörter konnten zusammen nur einmal beobachtet werden, also $n_{ab} = 1$. $sig_{Poisson}$ berechnet in dieser Situation einen Signifikanzwert von $-50,28$. sig_{lgl} hingegen beträgt $1999,86$. Beide Maße behandeln signifikant seltene Ereignisse offensichtlich anders. sig_{lgl} kann dementsprechend nicht zwischen signifikant häufigen und signifikant seltenen Ereignissen unterscheiden.

Werden die zehn Millionen unsignifikantesten Kookkurrenzen betrachtet (vgl. Abbildung 3.14), dann wachsen die Linien aus Abbildung 3.13 sukzessive zusammen.

[2]Distinktion bedeutet, dass ein Wort nur einmal pro Satz vorkommen darf.

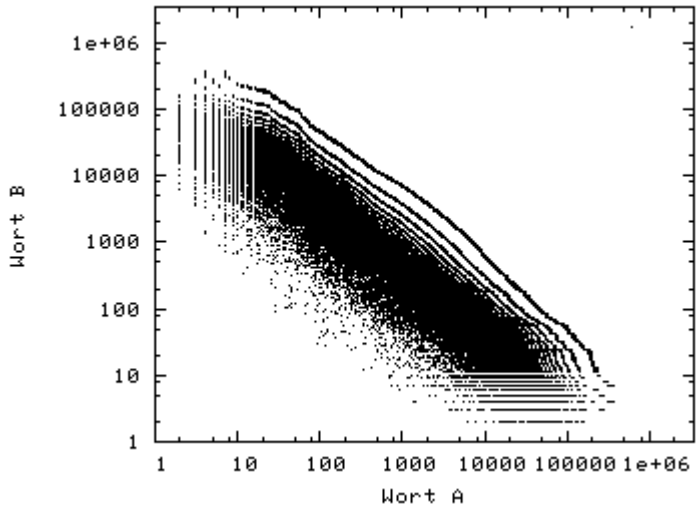

Abbildung 3.13: die eine Million *unsignifikantesten Kookkurrenzen* nach sig_{LGL}

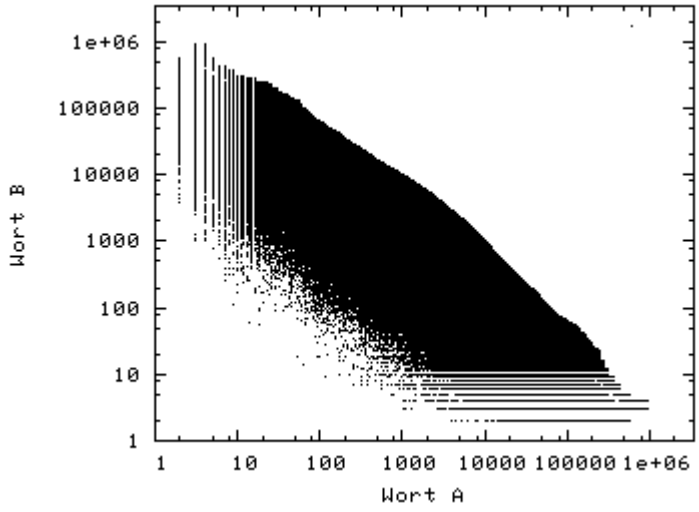

Abbildung 3.14: die zehn Millionen *unsignifikantesten Kookkurrenzen* nach sig_{LGL}

In Abbildung 3.15 sind die eine Million signifikantesten Kookkurrenzen abgebildet. Ein Vergleich mit dementsprechenden Plot des Poisson-Maßes zeigt, dass sich beide Maße bezüglich der Handhabung signifikant häufiger

Ereignisse nicht unterscheiden.

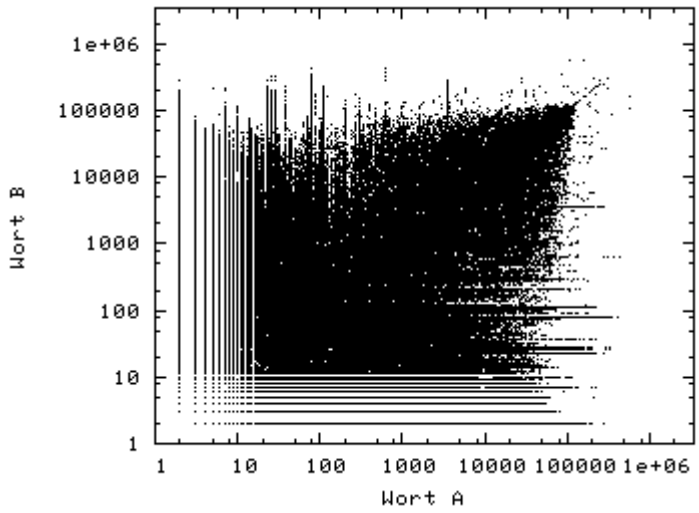

Abbildung 3.15: die eine Million *signifikantesten Kookkurrenzen* nach sig_{LGL}

3.2 Signifikanzen auf einem zufälligen Korpus

Die meisten der vorher vorgestellten Signifikanzmaße basieren auf der stochastischen Unabhängigkeit $p(n_{ab}) = p(n_a) * p(n_b)$. Dabei ist signifikant, wie stark für eine Kookkurrenz die stochastische Unabhängigkeit nicht erfüllt ist.

In diesem Zusammenhang stellt sich die Frage, inwiefern die Signifikanzmaße auch für einen Korpus funktionieren, der zufällig erstellt worden ist. Da die Kookkurrenzen rein zufällig und weder in einem semantischen Zusammenhang noch in einer festen syntaktischen Struktur stehen, ist zu erwarten, dass keine bzw. nur signifikante Kookkurrenzen zwischen niederfrequenten Wörtern extrahiert werden können, die aufgrund dessen signifikant sind, da bereits das einmalige gemeinsame Vorkommen über der zu erwartenden Frequenz liegt.

Um einen solchen Korpus zu erstellen, wurden strikt die Vorgaben des oben benutzten Fünf-Millionen-Satz-Korpus eingehalten. Dabei konnten die Parameter des *Zipfschen Gesetzes* (vgl. Kapitel 1.4) unter Verwendung der Formel 5 in [New05] bestimmt werden. Auf diese Weise wird ein zufälliger *zipfverteilter* Korpus erzeugt, der bezüglich der *Zipfverteilung* die gleichen

Parameter besitzt. Weiterhin wird darauf geachtet, dass für jeden Satz im Originalkorpus ein gleich langer im künstlich erzeugten Korpus produziert wird.

Ein Satz aus dem Originalkorpus lautet:

> Diese Spannen kamen nach Ansicht von Devisenhändlern einem Floaten der EWS-Währungen gleich .

Die Wörter dieses Satzes werden durch Wortnummern ersetzt. Dabei entspricht eine Wortnummer dem Rang eines Wortes aus dem Zipfschen Gesetz.

> 243 67884 841 39 805 9 37151 49 258769 3 120643 422 2

Schließlich wird ein gleich langer Satz erzeugt. In diesem Satz werden dabei korrespondierende Wortnummern erzeugt, die vom Zufallsgenerator produziert werden. Die Reihenfolge der Wortnummern ist somit rein zufällig.

> 3202 8383 25 14 131 28124 614 945 41 4 2625127 31 23

Unter Verwendung des Mappings von Wortnummer (Rang) auf ein bestimmtes Wort aus dem Originalkorpus kann aus dem eben zufällig erzeugten Satz die Wortfolge

> u. Zwölf eine des neuen Schreibweise Lage Nr. bei die Schmitzius es :

erzeugt werden.

Dieses Beispiel zeigt, dass der zufällig erzeugte Satz weder syntaktisch richtig noch semantisch sinnvoll ist. Auf diese Weise wurden insgesamt fünf Millionen Sätze generiert.

Abbildung 3.16 zeigt den *Rang-Frequenz*-Plot der Wortliste des zufällig erzeugten Korpus, der wesentlich besser einer Geraden entspricht als der korrespondierende Originalkorpus aus Abbildung 1.1.

Bei der Kookkurrenzanalyse auf einem zufällig erzeugten Korpus konnten drei Beobachtungen angestellt werden. Während aus dem korrespondierenden Originalkorpus rund 347 Millionen Kookkurrenzen extrahiert werden konnten, wurden auf dem Zufallskorpus 498 Millionen Kookkurrenzen beobachtet. Das liegt darin begründet, dass die Kookkurrenzen zufällig auftreten und damit kein semantischer Zusammenhang existiert. Dementsprechend treten besonders frequentere „Wörter“ mit deutlich mehr anderen Wörtern als im Originalkorpus auf.

In einer zweiten Analyse wurde untersucht, inwiefern es auf einem Zufallskorpus möglich ist, dass eine der beiden Wortfrequenzen kleiner ist als die Frequenz der Kookkurrenz. Dies geschieht immer dann, wenn innerhalb

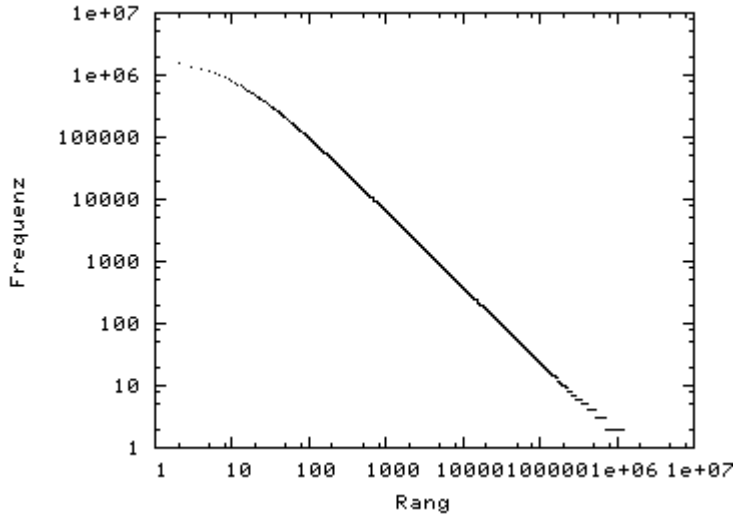

Abbildung 3.16: *Rang-Frequenz*-Plot der Wortliste eines zufällig erzeugten Korpus

eines Fensters ein Wort doppelt vorkommt und keine Distinktion der Wörter innerhalb dieses Fensters durchgeführt wird. Für das Berechnen der Signifikanz nach sig_{LGL} ist diese Bedingung notwendig, da sonst diverse Terme aus Formel 3.18 nicht berechnet werden können.

Auf dem Zufallskorpus konnten Kookkurrenzen, die die eben beschriebene Eigenschaft erfüllen, zwischen einem sehr seltenen und einem frequenten Wort beobachtet werden (vgl. Abbildung 3.17). Bei allen Kookkurrenzen, die in Abbildung 3.17 dargestellt sind, gilt $n_{ab} - \min(n_a, n_b) = 1$.

Der gleiche Test auf dem Originalkorpus zeigt (vgl. Abbildung 3.18), dass diese Bedingung wesentlich mehr Kookurrenzen erfüllen. Da im Zufallskorpus keine Kookkurrenzen auftreten, die diese Eigenschaft erfüllen und zwischen Wörtern auftreten, die mittel- und hochfrequent sind, kann dies auf semantische Zusammenhänge zurückgeführt werden. So werden Wörter aus Wendungen wie *mehr und mehr* und *viel zu viel* sehr häufig doppelt auftreten. Weiterhin sind Konstruktionen wie *... die Uhr , die ...* als kritisch zu sehen. In Kapitel 4 wird diesem Problem mit einer *spektralen Zerlegung einer Kookkurrenz* bezüglich des Abstandes der beiden Wörter entgegen getreten.

Abschließend werden Signifikanzen von Kookkurrenzen auf dem Zufallskorpus basierend auf den Maßen aus den Kapiteln 3.1.1 bis 3.1.6 berechnet. Die Ergebnisse sind in Tabelle 3.1 dargestellt.

Ein Ergebnis aus Tabelle 3.1 ist, dass die verteilungsbasierten Signifi-

Rang	sig_{Dice}	$sig_{Jaccard}$	sig_{LGL}	sig_{LGL2}	$sig_{Poisson}$	$sig_{LMI.}$	sig_{MI}
1	<1642787,2247440> *1,0*	<1642787,2247440> *1,0*	<16,16> *103953*	<16,16> *103953*	<16,16> *71902*	<16,16> *274820*	<1642787,2247440> *22,25*
2	<1249553,1222021> *1,0*	<1249553,1222021> *1,0*	<18,1> *22543*	<18,1> *22543*	<18,1> *12408*	<18,1> *118918*	<1249553,1222021> *22,25*
3	<1250280,3149420> *1,0*	<1250280,3149420> *1,0*	<7,13> *13848*	<7,13> *13848*	<7,13> *9241*	<7,13> *88214*	<1250280,3149420> *22,25*
4	<3270886,1652936> *1,0*	<3270886,1652936> *1,0*	<5,29> *8134*	<5,29> *8134*	<5,29> *5598*	<5,29> *53439*	<3270886,1652936> *22,25*
5	<1652938,2792008> *1,0*	<1652938,2792008> *1,0*	<47,2> *7372*	<47,2> *7372*	<24,9> *4722*	<24,9> *45876*	<1652938,2792008> *22,25*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9.999.996	<1499278,1> *0,0*	<1499278,1> *0,0*	<100,20773> *0,0*	<3152,326> *-4,61*	<168,6471> *-20,46*	<14846,19> *-10,73*	<168,6471> *-28,04*
9.999.997	<149926,29> *0,0*	<149926,29> *0,0*	<1001,3144> *0,0*	<2538,403> *-4,72*	<3152,326> *-21,04*	<15,16638> *-10,81*	<3152,326> *-29,37*
9.999.998	<134,571070> *0,0*	<134,571070> *0,0*	<1000,5548> *0,0*	<22,33505> *-4,98*	<22,33505> *-21,13*	<2713,76> *-11,86*	<22,33505> *-29,55*
9.999.999	<2499347,100> *0,0*	<2499347,100> *0,0*	<10003,221> *0,0*	<1460,330> *-5,12*	<2538,403> *-21,14*	<104,2481> *-11,92*	<2538,403> *-29,56*
10.000.000	<100000,11> *0,0*	<100000,11> *0,0*	<100015,47> *0,0*	<5,69013> *-6,18*	<5,69013> *-21,71*	<23,5880> *-12,33*	<5,69013> *-30,74*

Tabelle 3.1: signifikante und unsignifikante Kookkurrenzen eines Zufallskorpus (ausgewählt wurden die ersten zehn Millionen der 497 Millionen Kookkurrenzen, die aus dem Hash exportiert worden sind)

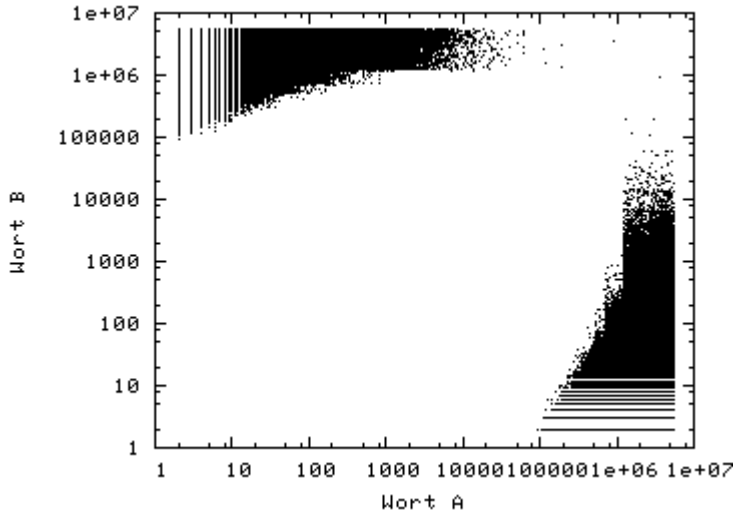

Abbildung 3.17: Kookkurrenzen eines *Zufallskorpus*, für die gilt: $n_{ab} > \min(n_a, n_b)$

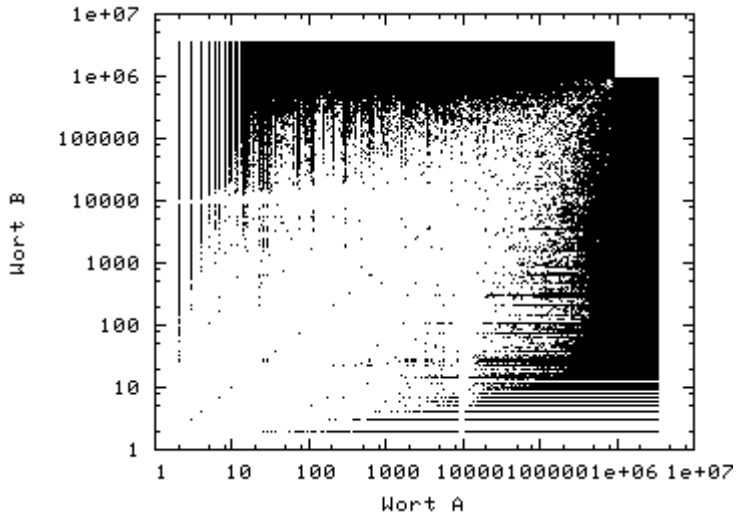

Abbildung 3.18: Kookkurrenzen eines *deutschen Korpus*, für die gilt: $n_{ab} > \min(n_a, n_b)$

kanzmaße $sig_{Poisson}$ sowie sig_{LGL} und die daraus ableitbaren Maße sig_{LGL2}[3]

[3] sig_{LGL2} entspricht dem numerischen Wert von sig_{LGL}. Jedoch wird aufgrund der stochastischen Unabhängigkeit das Vorzeichen entschieden.

und sig_{LMI} sowohl signifikant häufige als auch signifikant seltene Kookkurrenzen berechnen, obwohl diese Maße auf der stochastischen Unabhängigkeit basieren und dementsprechend auf einem Zufallskorpus keine signifikanten Kookkurrenzen berechnen dürften.

Tabelle 3.1 zeigt, dass für $sig_{Poisson}$, sig_{LGL}, sig_{LGL2} signifikant häufige Kookkurrenzen zwischen frequenten Wörtern auftreten. Auch nach einer Wortdistinktion innerhalb des Fensters bleiben diese Kookkurrenzen signifikant. Die signifikant seltenen Ereignisse für diese Maße liegen ebenfalls im frequenten Bereich.

Bei den Maßen sig_{Dice} und $sig_{Jaccard}$ kann beoboachet werden, dass genau wie beim Originalkorpus (vgl. Abbildungen 3.4) auf dem Zufallskorpus Kookkurrenzen als unsignifikant berechnet werden, die aus einem hochfrequenten und einem niederfrequenten Wort bestehen.

3.3 Zusammenfassung

Bei der Berechnung von Signifikanzen hat sich gezeigt, dass sich semantische Zusammenhänge am besten durch Maße wie sig_{LGL}, $sig_{Poisson}$ und sig_{LMI}, die auf der stochastischen Unabhängigkeit beruhen, berechnen lassen.

Als signifikant wird bei diesen Maßen eine Kookkurrenz erachtet, bei der die Beobachtung möglichst stark von der Erwartung abweicht. Bei einer Kookkurrenz zwischen zwei Wörtern, die 1.234- und 5.678mal in einem Korpus von fünf Millionen Sätzen vorkommen, würde kein statistischer Zusammenhang zwischen beiden Wörtern bestehen, wenn beide Wörter etwa $\frac{12.345*67.890}{5.000.000} = \frac{838.102.050}{5.000.000} \approx 168$mal zusammen auftreten. Bei einer Kookkurrenzfrequenz, die deutlich höher oder niedriger als 168 ist, ist dies statistisch auffällig.

Anders sieht das bei Kookkurrenzen zwischen zwei seltenen Wörtern aus. Insbesondere $sig_{Poisson}$, welches auf der *Poisson*-Verteilung (die Verteilung der seltenen Ereignisse) basiert, sollte mit seltenen Ereignissen umgehen können. Wird aber eine Kookkurrenz zwischen zwei Wörtern mit beispielsweise $n_a = 5$ und $n_b = 6$ betrachtet, die zusammen einmal auftreten, wird diese Kookkurrenz als signifikant berechnet und damit ein semantischer Zusammenhang zwischen diesen beiden Wörtern angenommen. Der Grund dafür liegt in der stochastischen Unabhängigkeit, die als Erwartung an diese Kookkurrenz eine „Frequenz“ von $6 * 10^{-6}$ berechnet. Tritt diese Kookkurrenz einmal auf, dann liegt die Beobachtung 166.667mal über der Erwartung. Folglich wird diese Kookkurrenz als signifikant erachtet.

Im Rahmen von *Medusa* werden Signifikanzen mit zwei Signifikanzmaßen

berechnet. Dabei wird dem eigentlichen Signifikanzmaß[4] ein *Cut-Off*-Maß vorgeschaltet, welches unabhängig vom Signifikanz-Maß eine Vorauswahl der Kookkurrenzen bestimmt. Als *Cut-Off*-Maße empfehlen sich einerseits sig_{freq} und andererseits sig_{Dice} sowie $sig_{Jaccard}$. Bei Benutzung von sig_{freq} werden beispielsweise alle Kookkurrenzen, die einmal aufgetreten sind, entfernt, so dass das Signifikanzmaß diese nicht fälschlicherweise als signifikant bestimmt.

Wird anstelle von sig_{freq} eines der Maße sig_{Dice} oder $sig_{Jaccard}$ benutzt, dann werden nicht alle Kookkurrenzen entfernt, die einmal aufgetreten sind, sondern nur Kookkurrenzen, die ein- oder zweimal zwischen einem frequenten und einem seltenen Wort beobachtet werden konnten (vgl. Abbildung 3.4), bei denen angenommen wird, dass es sich nur um ein Artefakt handelt.

Ferner sei an dieser Stelle auf die Ergebnisse aus dem Zufallskorpus verwiesen. Es konnte gezeigt werden, dass Kookkurrenzen, bei denen die Kookkurrenzfrequenz höher als die kleinere der beiden Wortfrequenzen ist, ausschließlich zwischen einem frequenten und einem seltenen Wort auftreten. Bei der Wahl von sig_{Dice} oder $sig_{Jaccard}$ als *Cut-Off*-Maß werden diese Kookkurrenzen herausgefiltert (vgl. Abbildungen 3.4 und 3.17). Kookurrenzen, die nach dem *Cut-Off* eine Kookkurrenzfrequenz besitzen, die über der kleineren der beiden Wortfrequenzen liegt, sind demnach nicht zufällig, sondern haben einen syntaktischen Hintergrund der entsprechenden Sprache.

Als Ergebnis wird festgehalten, dass sich Kookkurrenzen in drei Klassen einteilen lassen:

- Kookkurrenzen zwischen zwei frequenten Wörtern (*KLASSE 1*),
- Kookkurrenzen zwischen einem frequenten und einem seltenen Wort (*KLASSE 2*) sowie
- Kookkurrenzen zwischen zwei seltenen Wörtern (*KLASSE 3*).

Für die beiden ersten Klassen kann die Signifikanz und der damit verbundene semantische Zusammenhang mittels eines Signifikanz- und eines Cut-Off-Maßes, wie eben beschrieben, zuverlässig bestimmt werden.

Kritisch jedoch ist die *KLASSE 3*. Eingangs der Zusammenfassung dieses Kapitels wurde an einem Beispiel erklärt, dass das einmalige Auftreten zweier seltener Ereignisse bereits statistisch auffällig ist, unabhängig davon, ob es einen Zusammenhang zwischen beiden Wörtern gibt oder es sich um ein Artefakt handelt. Dieser Bereich ist letztlich deswegen als kritisch zu betrachten, da keines der Signifkanzmaße zuverlässig in diesem Bereich eine statistische Auffälligkeit bestimmen kann.

[4] In der Regel sind dies sig_{LGL}, $sig_{Poisson}$ oder sig_{LMI}.

Bei der Singifikanzbestimmung mittels eines Signifikanz- und eines Cut-Off-Maßes muss jedoch kritisch angemerkt werden, dass bei Kookkurrenzen aus der *KLASSE 3* nicht entschieden werden kann, ob es sich um ein Artefakt oder um einen semantischen Zusammenhang zwischen beiden Wörtern handelt. Als Ergebnis aus diesen Betrachtungen empfiehlt sich daher, die rein *statistik- bzw. mathematikbasierte Signifikanzbestimmung* in zwei Schritte aufzuteilen.

Im ersten Schritt werden für die erste Klasse die Signifikanzen mittels einer Kombination von Signifikanzmaß und Cut-Off-Maß, wie bereits beschrieben, bestimmt. Dabei werden Kookkurrenzen zwischen Wörtern aus dem niederfrequenten Bereich nicht berücksichtigt. Auf diesen Kookkurrenzen werden semantische Relationen wie beispielsweise die paradigmatische Relation *Kohyponym* bestimmt bzw. liegen bereits vorberechnet vor. Aus einem Beispielsatz,

> *Während eines Taiwanurlaubs wurde ein Tourist von einem Ebudu gebissen.*

der das Wort *Ebudu* enthält, welches bisher im Korpus nicht aufgetreten ist, kann so die Kookkurrenz *(gebissen, Ebudu)* extrahiert werden. An dieser Stelle sei auf das Kapitel 4 verwiesen. In diesem Kapitel werden Kookkurrenzen in ihr Spektrum bezüglich der Distanz der beiden Wörter einer Kookkurrenz zerlegt. Dabei wird festgestellt, dass *Ebudu* linker Nachbar (*Abstand=1*) von *gebissen* ist. Mittels der vorher berechneten Kookkurrenzen werden im zweiten Schritt Wörter bestimmt, die an der gleichen Stelle wie *Ebudu* stehen. Dies können Wörter wie *Pitbullrüde*, *Alligator* oder *Schlange* sein. Es sei angenommen, dass zwischen diesen Wörtern zuvor eine paradigmatische Relation bestimmt worden ist. Dann werden schließlich die Sätze, welche die Kookkurrenzen *(gebissen, Alligator,1)*, *(gebissen, Schlange,1)* und *(gebissen, Pitbullrüde,1)* auf Ähnlichkeit überprüft. Sollten sich aus diesen Sätze hinreichend viele Übereinstimmungen finden, dann wird die Kookkurrenz *(gebissen, Ebudu,1)* als signifikant erachtet. Dabei ist die Berechnung der Signifkanz nicht mehr mathematischen (*significance by measure*), sondern algorithmischen (*significance by algorithm* bwz. differenzierter *significance by semantic relation*) Ursprungs.

Eine Verfeinerung und Evaluierung dieses Ansatzes wird Gegenstand weiterer Forschung sein. Es sei an dieser Stelle auf ein Kernproblem bei der herkömmlichen Signifikanzberwertung hingewiesen. Church (vgl. [CH90]) hatte darauf hingewiesen, dass die *Mutual Information* nur auf kleinen Fenstern und für Kookkurrenzen, deren Wörter häufiger als fünfmal vorgekommen sind, benutzbar ist. Unter dem Gesichtspunkt, dass durch ein mathematisches Maß nicht alle „relevanten“ Kookkurrenzen extrahiert werden müssen,

wird die *Mutual Information* für einen solchen Ansatz durchaus sehr interessant, da die *Mutual Information* dem Quotienten aus der beobachteten und der erwarteten Wahrscheinlichkeit einer Kookkurrenz entspricht. Mögliche Probleme mit der *Mutual Information* bezogen auf Kookkurrenzen, die auf sehr seltenen Wörtern basieren, können somit umgangen werden. Letztlich konnte in diesem Kapitel gezeigt werden, dass auch sig_{LGL}, sig_{LMI} und $sig_{Poisson}$ die Signifikanz solcher Kookkurrenzen nicht zuverlässig berechnen können.

Im Rahmen dieser Arbeit wird für das Berechnen von Signifikanzen mit der *Local Mutual Information* gearbeitet. Es konnte in diversen Abbildungen gezeigt werden, dass das *Poisson-* sowie das *Log-Likelihood*-Maß im Wesentlichen das Gleiche wie die *Local Mutual Information* berechnen. Ein wesentlicher Unterschied zum *Log-Likelihood*-Maß besteht im Umgang mit signifikant seltenen Kookkurrenzen. Während beim *Log-Likelihood*-Maß sowohl signifikant häufige also auch seltene Kookkurrenzen mit einem positiv hohen Signifikanzwert ausgewiesen werden und damit keine Differenzierung zwischen beiden möglich ist, unterscheidet die *Local Mutual Information* diese beide Formen von Signifikanzen durch positiv hohe bzw. negativ kleine Signifikanzen.

Diese Unterscheidung wird beispielsweise im Anhang B[5] benötigt, in welchem Kookkurrenzen auf einem Werkstattkorpus eines Automobilherstellers berechnet werden. Eine Unterscheidung der Kookkurrenzen *(brake,tire)* bzw. *(brake,fuel pump)* wird damit möglich. Während erstere Kookkurrenz signifikant häufig auftritt, wird die Kookkurrenz *(brake,fuel pump)* signifikant selten auftreten. Beim *Log-Likelihood*-Maß würden solche Kookkurrenzen gemischt werden und die Daten wären dementsprechend bei einem Werkstattkorpus, bei welchem signifikante Kookkurrenzen zwischen Bauteilen berechnet werden sollen, weitestgehend unbrauchbar.

In Abgrenzung zum *Poisson*-Maß sei auf die Formel 3.13 verwiesen. Während bei der *Local Mutual Information* der Signifikanzwert positiv ist, wenn die Beobachtung der Kookkurrenz über der Erwartung liegt und negativ ist, wenn die Beobachtung einer Kookkurrenz unter der Erwartung bleibt, arbeitet das *Poisson*-Maß an dieser Stelle geringfügig unterschiedlich. Im Zähler der Formel 3.13 steht am Ende -1. Dies läßt auch Kookkurrenzen negativ werden, deren Beobachtung über der Erwartung liegt.

Weiterhin sei abschließend bei der *Local Mutual Information* noch einmal auf die Termgewichtung *tf*IDF* und deren Ähnlichkeit verwiesen.

[5] Anhang B ist nicht Teil der öffentlichen Diplomarbeit.

Kapitel 4

Distanzabhängige Kookkurrenzen

Bisherige Ansätze des Berechnens von Kookkurrenzen gehen von einer bestimmten Fenstergröße aus und alle extrahierbaren Kookurrenzen innerhalb dieses Fensters werden gezählt. Dabei ist die Wahl der Fenstergröße entscheidend für die Art der Beziehung, die extrahiert werden soll. Für jede Fenstergröße bedarf es schließlich einer separaten Berechnung.

Im Rahmen dieser Arbeit wurden bisher Kookkurrenzen auf Satzfensterbasis berechnet. In diesem Kapitel sollen Kookkurrenzen in ihr *Spektrum bezüglich des Abstandes* zwischen den beiden Wörtern zerlegt werden. Dabei wird eine Kookkurrenz nicht mehr nur beim gemeinsamen Auftreten, sondern beim gemeinsamen Auftreten im jeweiligen Abstand gezählt.

Das Ziel dieses Kapitels besteht aus zwei Teilen. Erstens soll überprüft werden, ob das Spektrum einer Kookkurrenz systematischer Natur ist (vgl. [dS01]). Wird beispielsweise das Spektrum der Kookkurrenz *(die,Uhr)* mit dem Spektrum von *(die,Distanz)* verglichen, so kann erwartet werden, dass sich die Spektren ähneln, wenn die Hypothese der Systematik korrekt ist. Ein zweites Ziel dieses Kapitels ist die Untersuchung, ob und wie sich verschiedene semantische Relationen in unterschiedlichen Abständen manifestieren.

Signifikanzmaße, wie sie in Kapitel 3 genannt worden sind, können zwar eine statistische Signifikanz berechnen, lassen dabei aber den Abstand der beiden Wörter völlig außer Acht. Beim Trainieren eines *Hidden Markov Modells* wird beispielsweise auf *Tri-*, *Bi-* und *Unigramme* mit der Begründung eines stark lokalen Zusammenhangs zurückgegriffen (vgl. [HQW06]). Bei der Berechnung von Satzkookkurrenzen bleibt dieser lokale Zusammenhang deutlich unberücksichtigt. Dementsprechend wird es ein weiteres Ziel dieser Arbeit sein, eine Modifikation herkömmlicher Signifikanzmaße vorzunehmen, so dass die Signifikanz nicht allein auf statistischen Annahmen, sondern auch

von linguistischem Wissen bezüglich der Satzstruktur und dementsprechend vom Abstand zweier Wörter abhängig wird.

Als Korpus wurden $35,7$ Millionen deutsche Sätze aus Zeitung benutzt. Aus diesem Korpus konnten $1,49$ Milliarden Kookkurrenzen auf Satzbasis berechnet werden. Nach einer Berechnung mit einer *spektralen Zerlegung* existieren $3,4$ Milliarden Kookkurrenzen. In Abbildung 4.1 ist die Verteilung dieser Kookkurrenzen bezüglich ihres Abstandes dargestellt.

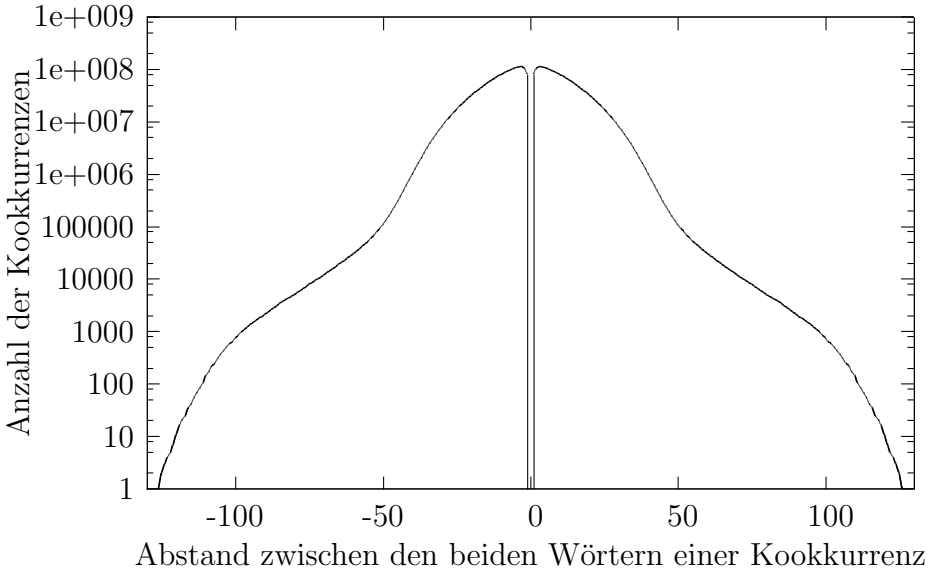

Abbildung 4.1: Verteilung der Kookkurrenzen von 36 Millionen deutschen Sätzen

Da das Berechnen von spektral zerlegten Kookkurrenzen nur dann sinnvoll ist, wenn die Kookkurrenzfrequenz hinreichend groß ist, werden in diesem Kapitel nur Kookkurrenzen zwischen den 100.000 häufigsten Wörtern berücksichtigt. Dies sind rund 2 von $3,4$ Milliarden der spektral zerlegten Kookkurrenzen. Die Mindestwortfrequenz liegt somit bei 205.

Im Rahmen dieser Arbeit wurde bisher für eine Kookkurrenz die Notation *(*$Wort_1$*,*$Wort_2$*)* benutzt. Diese Notation wird in diesem Kapitel um den Abstand zu *(*$Wort_1$*,*$Wort_2$*, dist)* erweitert. Dabei steht *dist* für den Abstand zwischen den beiden Wörtern $Wort_1$ und $Wort_2$. Ein Abstand von 3 bedeutet, dass $Wort_2$ in einem Satz drei Wörter nach $Wort_1$ beobachtet werden konnte. Ein negativer Abstand bedeutet, dass $Wort_2$ vor $Wort_1$ beobachtet werden konnte. Ein Distanz von 0 würde dementsprechend bedeuten, dass ein Wort mit sich selbst auftritt. Da dieser Fall nicht von Interesse ist, gibt es keine Kookkurrenzen im Abstand 0.

4.1 Distanzabhängiges Spektrum einer Kookkurrenz

Durch das Zerlegen einer Kookkurrenz in ihr Spektrum soll untersucht werden, ob sich für gleichartige Kookkurrenzen bezüglich POS-Tags[1] sich ähnliche Spektren ergeben. Dazu werden einerseits im Kapitel 4.1.1 distanzbasierte Kookkurrenzen auf dem Zufallskorpus aus Kapitel 3.2 betrachtet. Andererseits werden im Unterkapitel 4.1.2 einige ausgewählte Beispiele vorgestellt.

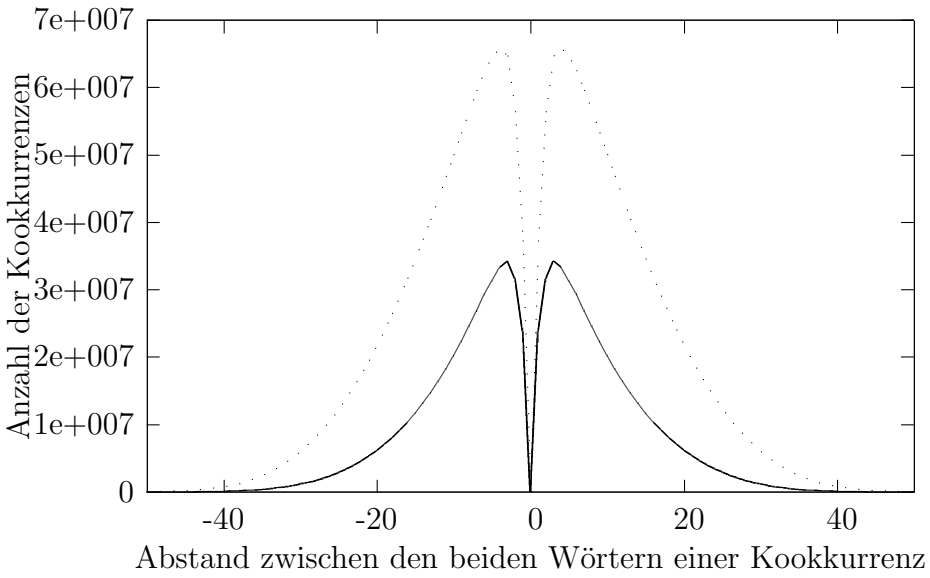

Abbildung 4.2: Verteilung der Kookkurrenzen (*gestrichelte Line*) und die entsprechend signifikanten Kookkurrenzen (*durchgezogene Linie*) im jeweiligen Abstand (zwischen den 100.000 häufigsten Wörtern)

Abbildung 4.2 zeigt die Verteilung von Kookkurrenzen zwischen den 100.000 häufigsten Wörtern aus einem natürlich-sprachlichen Korpus bezogen auf ihren Abstand. Die gestrichelte Linie entspricht dabei der Anzahl der Kookkurrenzen, die in dem jeweiligen Abstand beobachtet werden konnten. Die durchgezogene Line stellt die Anzahl der signifikanten Kookkurrenzen dar.

Abbildung 4.3 zeigt die kumulierte Verteilungsfunktion der signifikanten Kookkurrenzen aus Abbildung 4.2. Bei einem Abstand von 15 und kleiner werden dementsprechend rund 82% aller signifikanten Kookkurrenzen beobachtet. An dieser Stelle sei allerdings darauf hingewiesen, dass bei größeren

[1] *POS*: *Part-Of-Speech*. Ein POS-Tagger weist einer Wortform deren Wortklassen zu. Wortklassen können beispielsweise Substantive, Verben oder Adjektive sein.

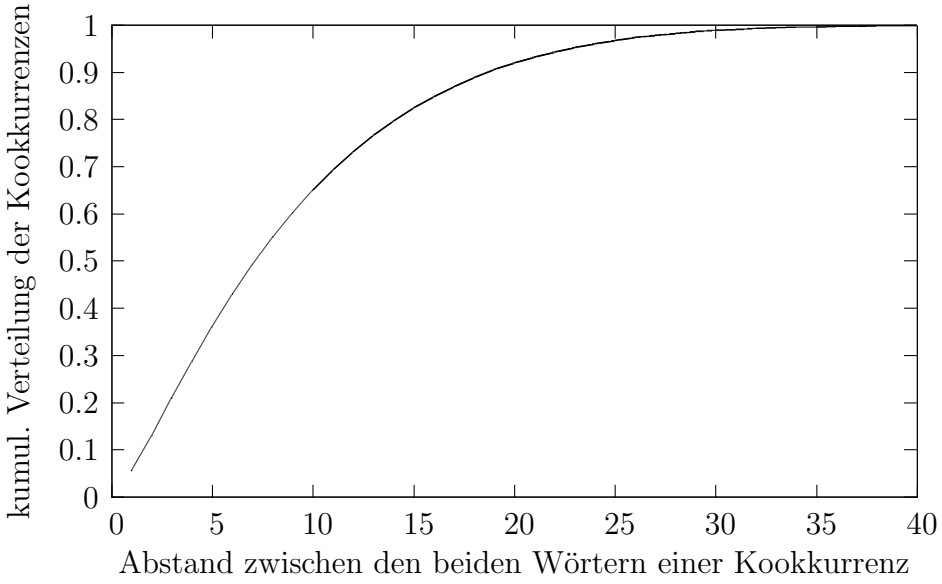

Abbildung 4.3: kumulierte Verteilungsfunktion signifikanter Kookkurrenzen aus Abbildung 4.2

Abständen von mehr als 60 nur noch signifikante Kookkurrenzen zwischen seltenen Wörtern gemessen werden können.

Bei der spektralen Zerlegung einer Kookkurrenz in verschiedene Abstände ergeben sich auch unterschiedliche Anforderungen an die Anzahl der Versuche n_{dist} im Abstand *dist*, die für das zu benutzende Signifikanzmaß von Bedeutung ist (vgl. Kapitel 3).

Eine Trennung in verschiedene n_{dist} für jeden Abstand ist notwendig, da aus einem Satz der Länge a genau $a-1$ *Kookkurrenz*-Tokens im Abstand 1 bestimmt werden können. Für einen Abstand von 2 sind es schließlich noch $a-2$. Da ein Korpus aus unterschiedlich langen Sätzen besteht, kann diese Folge nicht einfach fortgesetzt werden. Es ist aber offensichtlich: Je kleiner der Abstand, desto mehr Tokens können aus einem Satzfenster extrahiert werden.

Die Wahl von n_{dist} lässt sich aus der stochastischen Unabhängigkeit aus Formel 4.1 ableiten.

$$\frac{n_{ab}}{M} = \frac{n_a}{N} * \frac{n_b}{N} \tag{4.1}$$

Hierbei stehen M für die Anzahl der *Kookkurrenz*-Tokens im jeweiligen Abstand und N für die Anzahl der *Word*-Tokens. Daraus ergibt sich n_{dist} wie in Formel 4.2 dargestellt.

$$n_{dist} = \frac{N * N}{M} \tag{4.2}$$

Dabei fällt M wie in Abbildung 4.1 sukzessive ab. Im Abstand 1 gilt genau $n_1 = (s_l - 1) * n$, wobei n der Anzahl der Sätze und s_l der durchschnittlichen Satzlänge entspricht. Dies ist gleich der Anzahl der *Bigramme*, die aus einem Text mit n Sätzen extrahiert werden können.

Aus diesem Ansatz ergeben sich allerdings auch Probleme. Die Anzahl der *Kookkurrenz*-Tokens muss für jeden Abstand mitgezählt werden. Problematischer hierbei ist allerdings die Wahl von N. Werden für N die Anzahl der *Wort*-Tokens eines Korpus benutzt, dann führt dies bei größeren Abständen bzw. bei nieder- und mittelfrequenten Kookkurrenzen zu Problemen. Hierbei werden Kookkurrenzen ab einem Abstand von etwa 20 immer als signifikant angesehen. Das Problem liegt in der Benutzung einer Frequenzliste für die Wörter. Es müsste für jeden Abstand eine Frequenzliste der Wörter erstellt werden. Dadurch könnte schließlich mit einem N_{dist} für jeden Abstand gerechnet werden.

Im Rahmen dieser Arbeit soll im Kapitel 4.1.1 gezeigt werden, dass der Grundgedanke prinzipiell richtig ist. Dabei wird auf einem Spektrum der häufigsten Kookkurrenzen aus dem Zufallskorpus (vgl. Kapitel 3.2) zurückgegriffen.

Für die Spektren auf einem reellen Korpus wird an dieser Stelle eine Vereinfachung vorgenommen. Die Ergebnisse haben bestätigt, dass $n_{dist} = (s_l - 1) * n$ eine sinnvolle Approximation ist. Dabei wird n_1 für alle anderen n_{dist} angenommen.

4.1.1 Zufallskorpus

In Kapitel 3.2 wurden Kookkurrenzen auf einem zufälligen, aber zipfverteilten Korpus berechnet. Dabei konnte gezeigt werden, dass es zahlreiche signifikante Kookkurrenzen gibt. In diesem Kapitel soll auf die nach sig_{LMI} signifikanteste Kookkurrenz *(16,16)* bezüglich deren Abstandsspektrum näher eingegangen werden.

In Abbildung 4.4 ist das Spektrum des Abstandes der Kookkurrenz *(16,16)* dargestellt. Die gestrichelte Linie entspricht dabei der Frequenz, die nach der stochastischen Unabhängigkeit erwartet werden kann.

Abbildung 4.4 zeigt, dass die Kookkurrenz *(16,16)* im Abstand -1 und 1 genauso oft beobachtet werden konnte wie es auch zu erwarten war. Bei größeren Abständen in beiden Richtungen fällt die beobachtete Frequenz deutlich unter die erwartete Frequenz. Dies liegt hauptsächlich darin begründet, dass die Satzlänge nicht einheitlich ist.

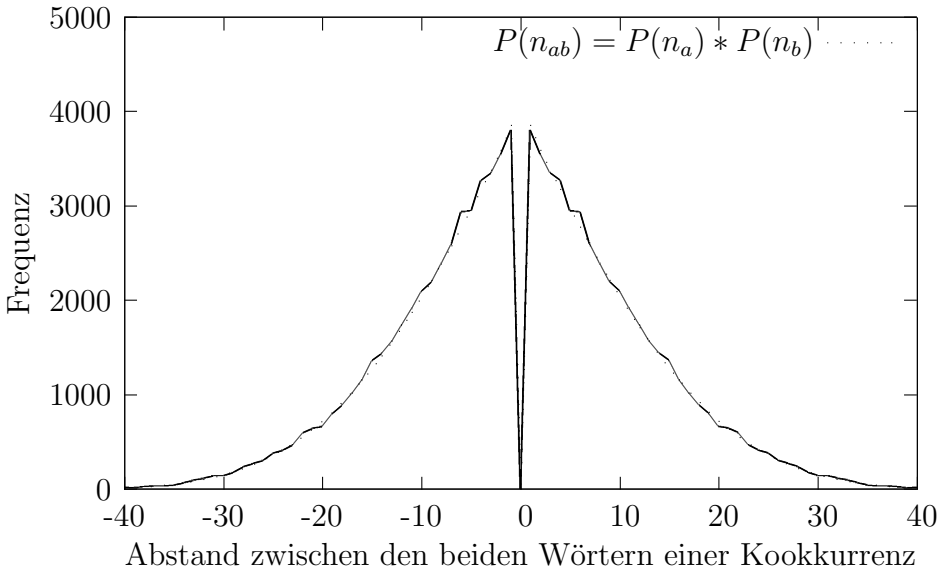

Abbildung 4.4: Frequenzspektrum der Kookkurrenz *(16,16)* aus Tabelle 3.1

Das Überprüfen der zehn signifikantesten Kookkurrenzen aus 3.1 hat gezeigt, dass diese Kookkurrenzen allesamt das gleiche Ergebnis geliefert haben:

- In allen Abständen entspricht die beobachtete der erwarteten Frequenz.
- Die Frequenzen im Abstand $-d$ und d sind nahezu gleich groß. Dadurch ist die Frequenzverteilung aus der Abbildung 4.4 bezogen auf die Senkrechte im Nullpunkt der x-Achse symmetrisch.

Zusammenfassend kann festgehalten werden, dass auf einem Zufallskorpus ausschließlich Kookkurrenzen als signifikant erachtet werden, die auf Satzbasis ohne spektrale Zerlegung berechnet werden. Werden die Satzkookkurrenzen in ihr Spektrum zerlegt, dann können solche Artefakte herausgefiltert werden. Bei der Berechnung von Nachbarschaftskookkurrenzen hingegen werden keine signifikanten Kookkurrenzen berechnet (vgl. Abbildung 4.4).

4.1.2 Ausgewählte Beispiele eines realen Korpus

Nachdem im Kapitel 4.1.1 in der Abbildung 4.4 das Spektrum einer zufällig erzeugten Kookkurrenz dargestellt worden ist, sollen in diesem Kapitel einige ausgewählte Spektren von Kookkurrenzen dargestellt werden.

Als Korpus wurde der *de*-Korpus des Wortschatz-Projektes (vgl. [Qua98]) mit rund 35 Millionen Sätzen benutzt. Ziel dieses Kapitels ist es die prinzi-

pielle Machbarkeit einerseits und den Nutzen einer *spektralen Zerlegung von Kookkurrenzen* bezüglich des Abstandes der beiden Wörter andererseits zu zeigen. Als Auswahl für die dargestellten Spektren dienen dabei Wortklassen. So werden beispielsweise Kookkurrenzen zwischen *Artikeln* und *Substantiven*, zwischen *Adjektiven* und *Substantiven*, *Verben* und *Substantiven* sowie zwischen *Substantiven* in ihren jeweiligen Spektren dargestellt. Abschließend werden noch Kookkurrenzen zwischen Vornamen und *die* bzw. *der* mit dem Ziel der Bestimmung des zum Namen passenden Geschlechts betrachtet.

In Abbildung 4.5 ist das Spektrum der Kookkurrenz *(die,Kunst)* dargestellt. Ein positiver Abstand zwischen *(die,Kunst)* bedeutet dabei, dass *Kunst* mit dem jeweiligen Abstand in rechter Nachbarschaft zu *die* steht. Ein Abstand von 1 bedeutet, dass auf *die* unmittelbar *Kunst* folgt. Abbildung 4.5

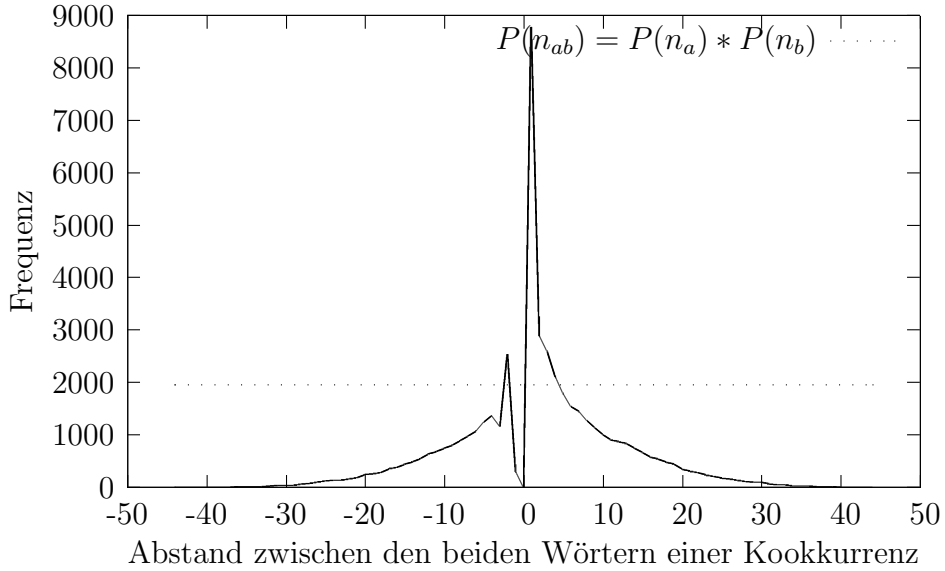

Abbildung 4.5: Spektrum der Kookkurrenz *(die,Kunst)*

zeigt, dass *Kunst* mit einem Abstand von 1 bis 4 auf *die* auffällig folgt. Hierbei steht die Kookkurrenz *(die,Kunst)* in einer *Artikel-Substantiv*-Beziehung. Weiterhin gibt es eine Peak bei einem Abstand von -2. Dies stellt eine *Relativpronomen-Substantiv*-Beziehung dar. Ein solches Spektrum läßt sich signifikant auch bei anderen Substantiven reproduzieren.

Da beispielsweise *die* und *der* zwei Wörter sind, die in einem deutschen Korpus sowohl als *Artikel* als auch als *Relativpronomen* benutzt werden können, werden, durch Wörter wie diese, Kookkurrenzen gezählt, deren Kookkurrenzfrequenz höher als die minimale Wortfrequenz der beiden Wörter ist.

Abbildung 4.5 zeigt, dass beispielsweise jedes Auftreten der Kookkurrenz *(die,Kunst)* gar nicht gezählt werden muss, da die syntaktische Struktur eines Satzes nur die beiden eben beschriebenen Möglichkeiten zulässt. Um auf diese Weise eine Distinktion der Wörter innerhalb des zu untersuchenden Fensters zu vermeiden, sei an dieser Stelle auf die Möglichkeit hingewiesen, dass für solche Kookkurrenzen lediglich im Bereich $[-3, 6]$ entsprechende *Kookkurrenz*-Tokens berücksichtigt werden müssen. Tokens außerhalb dieses Bereichs können mittels einer solchen '*Soft-Distinktion*' als Rauschen aus der Kookkurrenzfrequenz herausgefiltert werden.

Abbildung 4.6 zeigt das Spektrum der Kookkurrenz *(die,Uhr)*. Es ist deutlich zu sehen, dass bei jedem Abstand diese Kookkurrenz deutlich unter der Erwartung zurückbleibt (gestrichelte Linie). Dies liegt daran, dass es zwei verschiedene Bedeutungen von *Uhr* gibt. Einerseits kann der physische Zeitmesser damit gemeint sein. In diesem Zusammenhang wird der Artikel *die* benutzt. Andererseits kann *Uhr* auch als Zeitangabe wie beispielsweise *20 Uhr* benutzt werden. In diesem Zusammenhang wird der Artikel *die* nicht benutzt. Da letztere Bedeutung öfter im Korpus benutzt wird, geht erstere im Spektrum unter.

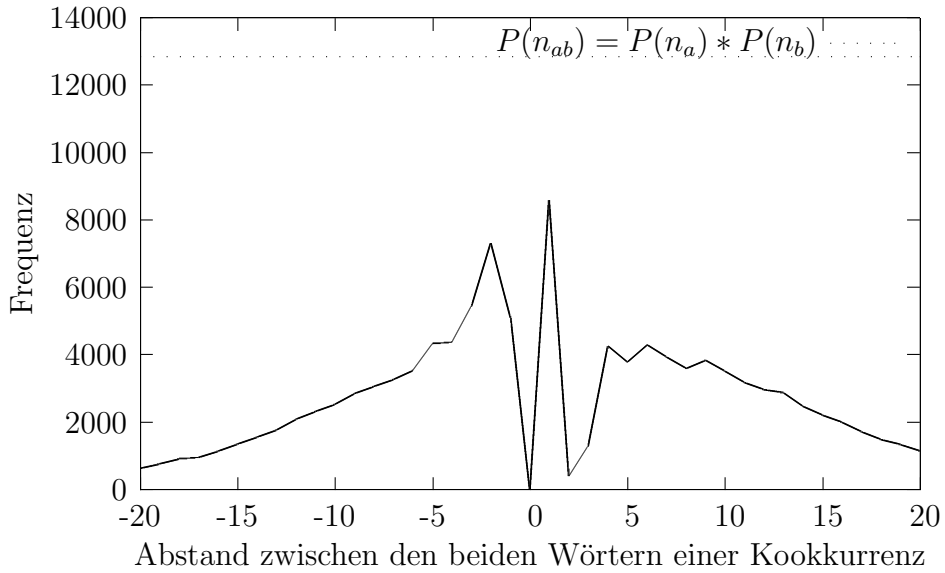

Abbildung 4.6: Spektrum der Kookkurrenz *(die,Uhr)*

Bei einer Aufspaltung der Wortfrequenz von *Uhr* auf die beiden Kontexte, würde die Erwartung dieser Kookkurrenz gesenkt werden und somit wieder das Verhalten aus Abbildung 4.5 sichtbar werden. Selbst bei einer

gleichmäßigen Aufteilung der Wortfrequenz von *Uhr* auf die beiden Bedeutungen würde die erwartete Frequenz auf etwa 6.500 fallen. Dann wäre die Kookkurrenz *(die,Uhr)* in den Abständen -2 und 1 signifikant.

Substantive sind bedeutungstragende Elemente eines Satzes. Daher wurde überprüft, inwiefern sich das Spektrum einer Kookkurrenz zwischen Substantiven zu Kookkurrenzen anderer Wortarten unterscheidet.

Im Gegensatz zum Spektrum der Kookkurrenz *(die,Uhr)* ist das Spektrum der signifikanten Abstände der Kookkurrenz *(Ausstellung,Uhr)* deutlich größer (vgl. Abbildung 4.7). Bei anderen Kookkurrenzen zwischen Substantiven ist ein ähnlich breites Spektrum bezüglich der Distanz zu beobachten.

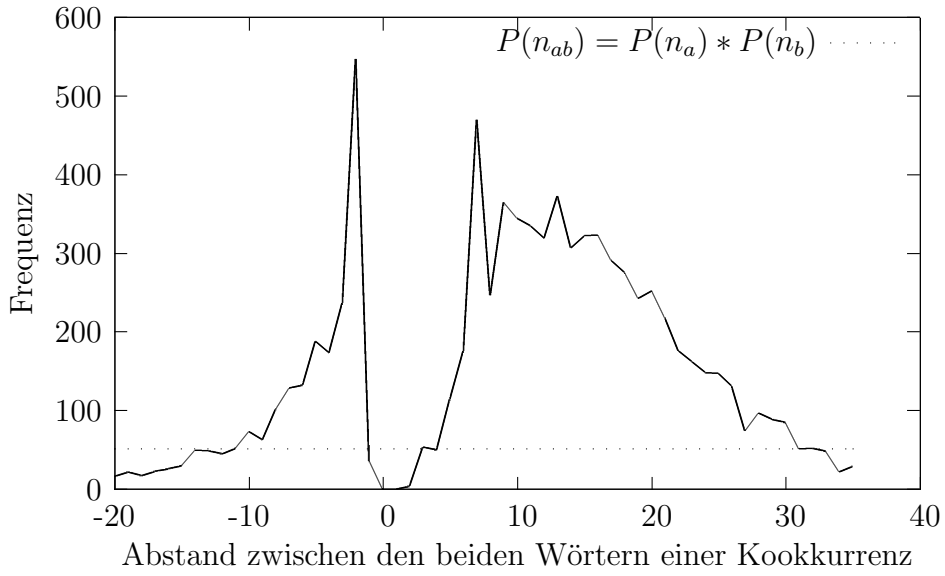

Abbildung 4.7: Spektrum der Kookkurrenz *(Ausstellung,Uhr)*

Abbildung 4.7 zeigt, dass die Kookkurrenz zwischen zwei Substantiven nicht symmetrisch ist. So gilt bei der Kookkurrenz *(Ausstellung,Uhr)*, dass *Uhr* vorzugsweise in einem Abstand von 3 bis 30 nach *Ausstellung* folgt und *Uhr* nur in einem Abstand von -10 bis -2 vor *Ausstellung* auftritt. Die Gründe dafür liegen in der deutschen Satzstruktur. Dabei treten *Ausstellung* und *Uhr* in unterschiedlichen Satzteilen auf. Während *Ausstellung* bevorzugt im Subjekt eines solchen Satzes steht, kann *Uhr* vorwiegend im Objekt bzw. im darauf folgenden Relativsatz beobachtet werden. Schließlich können dadurch auch die relativ hohen Abstände zwischen diesen beiden Wörtern begründet werden.

Ein geeignetes Prädikat für diese Kookkurrenz ist beispielsweise *beginnt*.

Das Spektrum einer Kookkurrenz wie *(beginnt,Uhr)* zwischen einem Verb und einem Substantiv ist in Abbildung 4.8 dargestellt. Für diese spezielle Kookkurrenz zeigt sich, dass *Uhr* tendenziell häufiger rechts vom Verb *beginnt* innerhalb eines Satzes beobachtet werden kann. Wird das Wort *Uhr* links von *beginnt* beobachtet, dann geschieht dies hauptsächlich unmittelbar nebeneinander. Genau wie bei Kookkurrenzen zwischen Substantiven ist auch bei *Verb-Substantiv*-Kookkurrenzen der Satzbau im Deutschen deutlich spürbar.

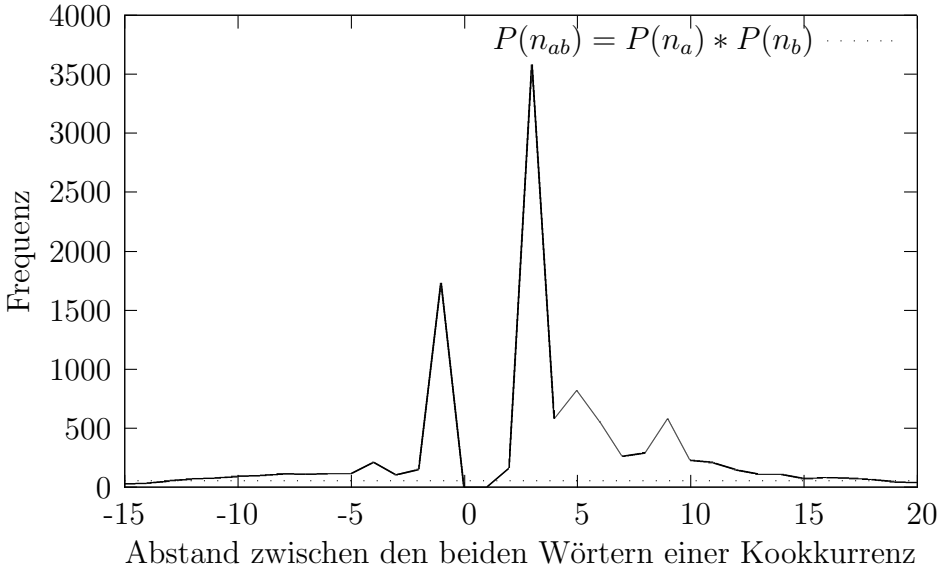

Abbildung 4.8: Spektrum der Kookkurrenz *(beginnt,Uhr)*

Bei Kookkurrenzen zwischen einem Adjektiv und einem Substantiv ist zu erwarten, dass das Adjektiv in einem relativ kleinen Fenster vor dem Substantiv beobachtet werden kann. Für eine Kookkurrenz *(alter,Mann)* ist das Spektrum in Abbildung 4.9 dargestellt.

Abbildung 4.9 zeigt, dass die Kookkurrenz *(alter,Mann)* in den Abständen 1 und 3 signifikant über der Erwartung beobachtet werden konnte. Hingegen konnte *alter* in keinem Abstand rechts von *Mann* beobachtet werden. Auch bei diesem Beispiel wird deutlich, dass die Satzstruktur der deutschen Sprache genau dies erwarten ließ.

Bisher wurden Spektren zwischen speziellen Wortklassen vorgestellt. Der letzte Teil dieses Kapitels beschäftigt sich mit Vornamen. Im Rahmen der Evaluierungen zu diesem Kapitel konnte beobachtet werden, dass sich auch für Vornamen spezifische Profile ergeben.

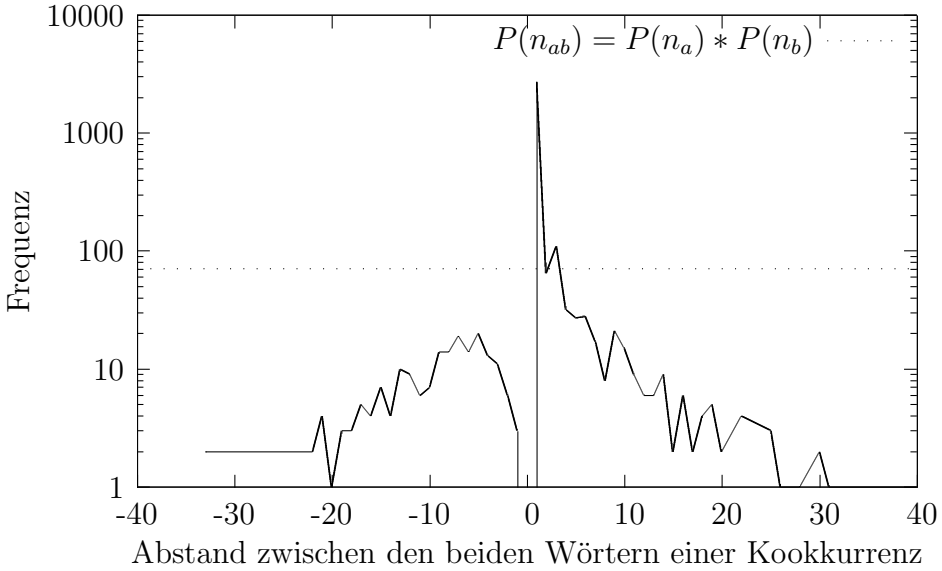

Abbildung 4.9: Spektrum der Kookkurrenz *(alter,Mann)* - logarithmisch skaliert

Um das Geschlecht eines Namens festzustellen dient einerseits der *Artikel* und andererseits das *Relativpronomen*. Dies gestaltet sich aber insofern als kritisch, als dass der weibliche Artikel im *Genitiv* und *Dativ* ebenfalls *der* ist. Im Rahmen dieser Evaluierungen werden nachfolgend beispielhaft die Spektren der beiden Namen *Katrin* und *Karsten* dargestellt. Dabei wurde sich auf die beiden Artikeln *der* und *die* beschränkt.

Die Abbildungen 4.10 und 4.11 zeigen die Spektren eines weiblichen (*Katrin*) bzw. eines männlichen (*Karsten*) Namens mit dem jeweiligen Artikel bzw. Relativpronomen.

Aus beide Grafiken können zwei signifikante Eigenschaften entnommen werden. Erstens gibt es drei Abstände (-3, 2 und 3), in denen der Vorname mit *die* bzw. *der* signifikant über der Erwartung beobachtet werden kann. Die positiven Abstände entsprechen dabei dem Artikel, der sich auf ein Substantiv wie *Weltmeisterin* oder *Weltmeister*, der dem Vornamen vorgeschaltet ist, bezieht. Der Abstand -3 entspricht dem dazugehörigen Relativpronomen. Ein Abstand von -3 heißt, dass zwei andere Wörter zwischen den beiden Wörtern einer Kookkurrenz stehen. Bei Vornamen sind dies im Konkreten der Nachname sowie das Komma vor dem Relativpronomen. Daher ist auch ein Abstand von -3 logisch nachvollziehbar. Zweitens ist in beiden Fällen der Peak des Relativpronomens deutlich höher als der Peak des Artikels. Dieses signifikante *Pattern* läßt sich über große Teile einer Namensliste beobachten.

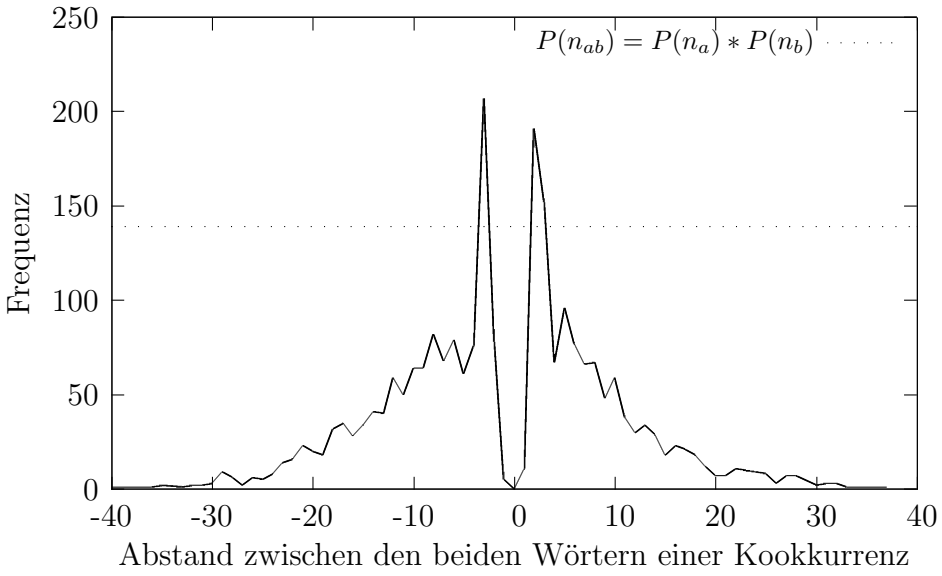

Abbildung 4.10: Spektrum der Kookkurrenz *(die,Katrin)*

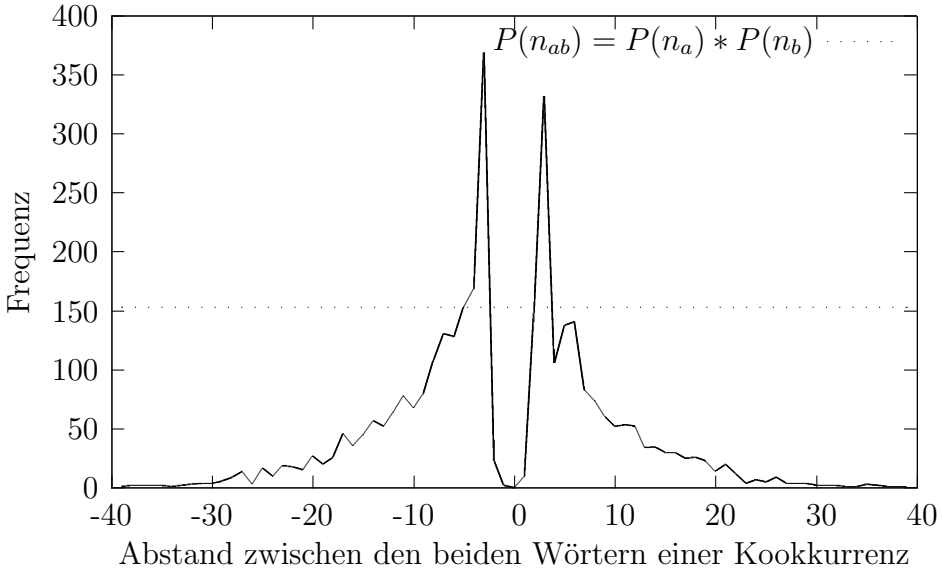

Abbildung 4.11: Spektrum der Kookkurrenz *(der,Karsten)*

In den Abbildungen 4.12 und 4.13 sind die Spektren der Kookkurrenzen zwischen einem Vornamen mit dem Artikel des jeweils anderen Geschlechtes dargestellt.

Beiden Abbildungen zeigen nicht mehr die signifikanten Peaks aus den Abbildungen 4.10 und 4.11 und deren Verhältnis zueinander. Aus Abbildung 4.12 ist ersichtlich, dass nur ein Peak beim Abstand von 3 signifikant über der Erwartung beobachtet werden kann. Der Grund dafür liegt im Genitiv und Dativ eines weiblichen Substantives.

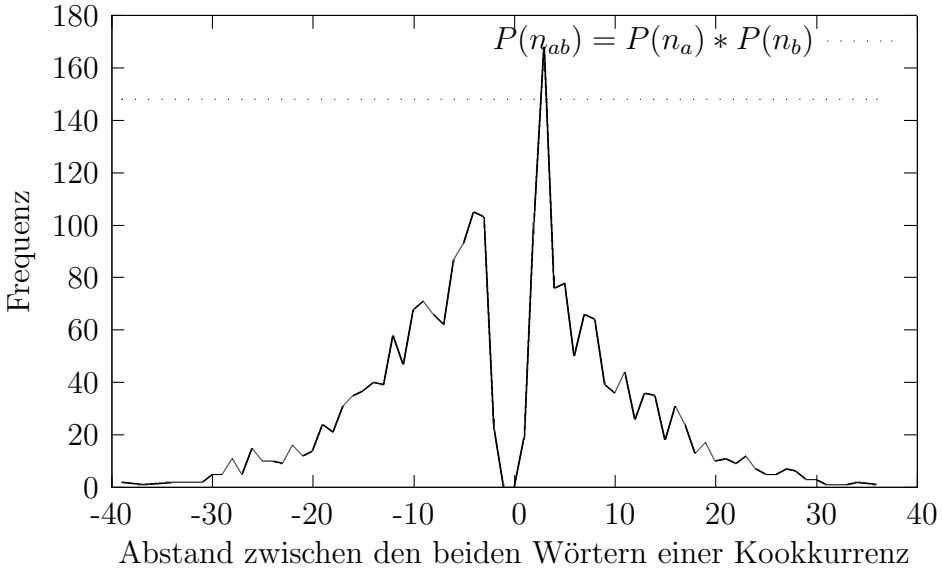

Abbildung 4.12: Spektrum der Kookkurrenz *(der,Katrin)*

Im Gegensatz zu Abbildung 4.12 kann bei einem männlichen Vornamen mit dem Artikel *die* in keinem Abstand eine signifikante Kookkurrenz beobachtet werden.

Unberücksichtigt bleibt an dieser Stelle das Verhalten bei Vornamen, die sowohl Männer- als auch Frauennamen sein können bzw. Namen wie *Paris* und *San Diego*, die Städtenamen entsprechen. Eine systematische Untersuchung ist jedoch nicht Bestandteil dieser Arbeit und wird Gegenstand weiteren Forschungsinteresses sein.

4.1.3 Semantische Relationen in unterschiedlichen Abständen

In Kapitel 4 wurden bisher einige Spektren zwischen ausgewählten Kookkurrenzen vorgestellt. In diesem Unterkapitel sollen die signifikanten Kookkurrenzen bei einem bestimmten Abstand bezüglich semantischer Relationen (siehe Kapitel 1.3) näher untersucht werden.

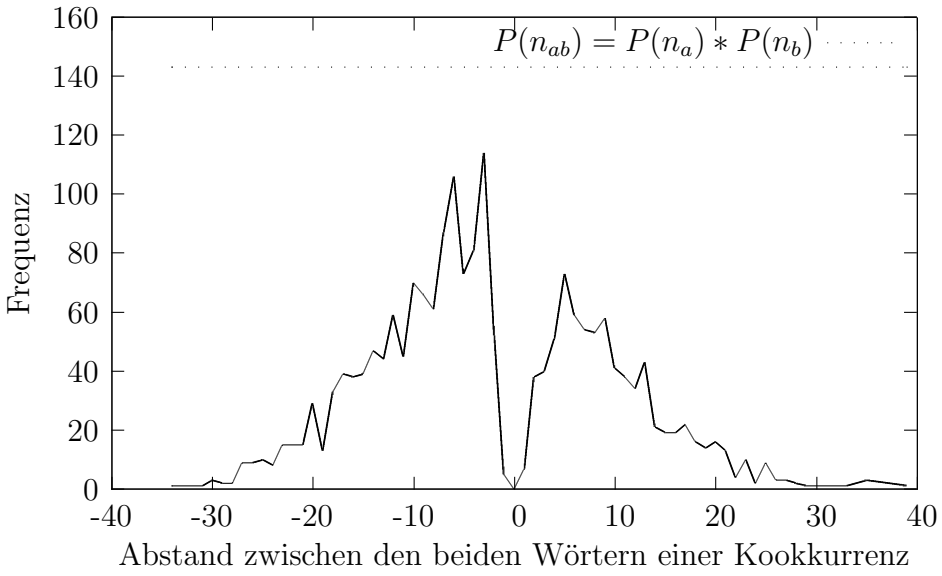

Abbildung 4.13: Spektrum der Kookkurrenz *(die,Karsten)*

Für diese Berechnungen wurde der eingangs des Kapitels vorgestellte Korpus *de* mit 35,7 Millionnen Sätzen benutzt. Es wurden Satzkookkurrenzen in ihrem jeweiligen Abstand gezählt. Anschließend wurden die signifikanten Kookkurrenzen eines Abstandes in einer Datei gespeichert. Auf diese Weise sind 238 Dateien entstanden. Im Rahmen dieser Arbeit werden lediglich die Ergebnisse der Abstände -20 bis 20 vorgestellt. Als Signifikanzmaß wurde die *Local Mutual Information* aus Kapitel 3 mit einem Schwellwert von 3 benutzt.

Es wurde bei der Berechnung keine Distinktion gemacht, da dies durch die spektrale Zerlegung nicht notwendig ist. Somit ist n_{ab} die Anzahl des gemeinsamen Auftretens zweier Wörter in einem bestimmten Abstand. n_a und n_b entsprechen den Wortfrequenzen der beiden Wörter im Korpus. Die Anzahl der Versuche n wurde aus der stochastischen Unabhängigkeit abgeleitet (siehe Formel 4.2).

Jede Datei, die auf diese Weise entstanden ist und die die entsprechenden signifikanten Kookkurrenzen in einem bestimmen Abstand beinhaltet, wurde im letzten Schritt mit einer Datenbank von annotierten Kookkurrenzen abgeglichen (vgl. [SBW05]). Diese Datenbank ist Teil des Wortschatzprojektes (vgl. [Qua98]) und wurde in einem Zeitfenster von zwei Jahren manuell annotiert. Annotiert wurde dabei die semantische Relation, die für die Kookkurrenz gilt. Beispielsweise wurde für eine Kookkurrenz *(T-Shirt,Hemd)* die

semantische Relation *N N Kohyponym* annotiert.

Zur Evaluierung wurden die zehn signifikantesten Kookkurrenzen pro Wort benutzt. Als Evaluierungsmaß wurde eine für diese Zwecke modifizierte Version der *Mean Average Precision* aus Formel 4.3 benutzt (vgl. [Wik05]).

$$MAP = \frac{\sum_{i=1}^{n} P(r) * rel(r)}{\text{Anzahl relevanter Dokumente}} \tag{4.3}$$

r repräsentiert den Rang der Kookkurrenz. $rel(r)$ ist eine binäre Funktion, die über die Relevanz der Kookkurrenz im Rang r entscheidet. $P(r)$ berechnet die *Precision*.

Durch die *Mean Average Precision* ist das Ranking der Kookkurrenzen von Bedeutung. Sind beispielsweise die ersten drei Kookkurrenzen relevant, ergibt sich aus Formel 4.3

$$MAP = \frac{\frac{1}{1} + \frac{2}{2} + \frac{3}{3}}{3} = 1 \tag{4.4}$$

Sind jedoch die drei relevanten Kookkurrenzen auf den Rängen 8, 9 und 10, ergibt sich eine *Mean Average Precision* von

$$MAP = \frac{\frac{1}{8} + \frac{2}{9} + \frac{3}{10}}{3} = \frac{\frac{233}{360}}{3} = \frac{233}{1080} = 0,2157 \tag{4.5}$$

Der Wert des Signifikanzmaßes für eine Kookkurrenz wird letzlich nur für das Ranking benutzt und spielt weiterführend keine Rolle mehr.

Im Folgenden werden die Ergebnisse für *syntagmatische*, *paradigmatische*, *hierarchisch-paradigmatische* Relationen sowie von *Derivaten* vorgestellt. Im Anhang A sind die ausführlichen Ergebnisse für die Abstände -20 bis 20 abgebildet. Neben den hier behandelten Relationen werden im Anhang A die Ergebnisse von 31 semantischen Relationen dargestellt, die unter anderem für die Berechnung von *syntagmatischen*, *paradigmatischen*, *hierarchisch-paradigmatischen* Relationen sowie von *Derivaten* notwendig gewesen sind. Aus Komplexitätsgründen wurde auf das Darstellen dieser Ergebnisse weitestgehend verzichtet. Da sie aber einmal berechnet sind, können sie als Grundlage für andere Arbeiten dienen.

Abbildung 4.2 zeigt die Verteilung der signifikanten Kookkurrenzen im jeweiligen Abstand. Bei einem Abstand von -15 bis 15 existieren teilweise deutlich mehr als eine Million signifikante Kookkurrenzen pro Abstand. In der oben beschriebenen Annotationsdatenbank existieren jedoch nur 414.878 Zuordnungen zwischen einer Kookkurrenz und der korrespondierenden *semantischen Relation*. Es sei daher darauf hingewiesen, dass es sehr viele Kookkurrenzen gibt, die zwar signifikant häufig aufgetreten sind und denen damit ein semantischer Zusammenhang unterstellt wird, die aber nicht in

der Datenbank annotiert sind und für die Betrachtungen ignoriert werden. Sie sind aber deswegen keineswegs wertlos.

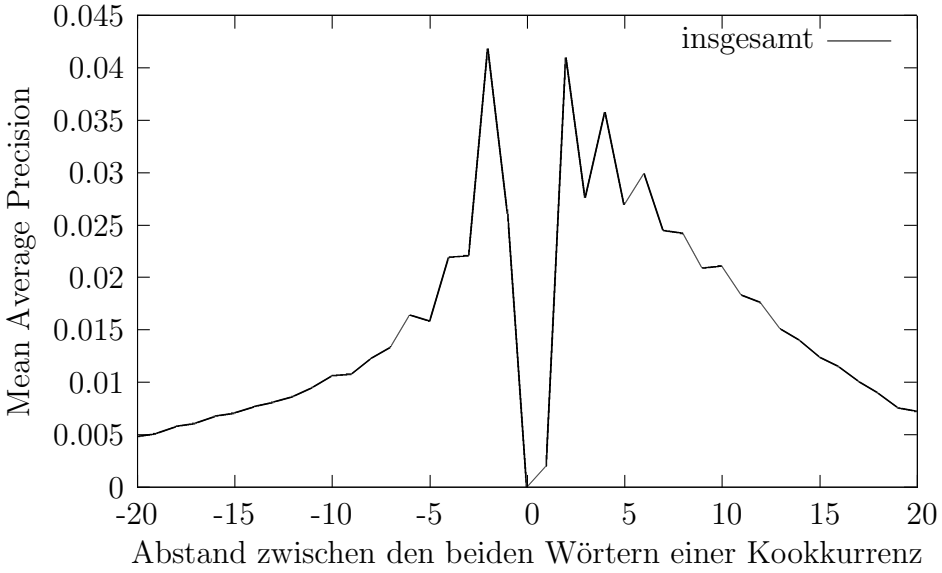

Abbildung 4.14: Verteilung der Summe aller semantischen Relationen über ihre Abstände

Abbildung 4.14 zeigt die Verteilung der Summe aller semantischen Relationen über dem Spektrum des Abstandes. Der y-Wert entspricht der *Mean Average Precision*. Der Plot zeigt einerseits, dass mit größer werdendem Abstand weniger semantische Relationen bzw. semantische Relationen, die auf einen niedrigeren Rang gesetzt worden sind, gefunden werden können. Andererseits zeigt der Plot auch, dass die *Mean Average Precision* für Kookkurrenzen in rechter Nachbarschaft tendenziell höher ist als für Kookkurrenzen in linker Nachbarschaft.

In den Abständen -1 und 1 können in beiden Fällen etwas mehr als 20.000 Kookkurrenzen in der Annotationsdatenbank gefunden werden. Mit zunehmenden Abständen zwischen den beiden Wörtern nimmt diese Zahl deutlich ab, so dass bei den Abständen -20 und 20 nur noch etwas mehr als 4.000 Kookkurrenzen in der Datenbank gefunden werden.

Eine Differenzierung der Abbildung 4.14 ist in Abbildung 4.15 dargestellt. Hierbei wurde aus der Verteilung der Summe aller Relationen die *syntagmatische*, *paradigmatische* und *hierarchisch-paradigmatische* Relation separat betrachtet.

Abbildung 4.15 zeigt, dass einerseits für die drei Relationen die *Mean Average Precision* mit steigender Entfernung der beiden Wörter abnimmt.

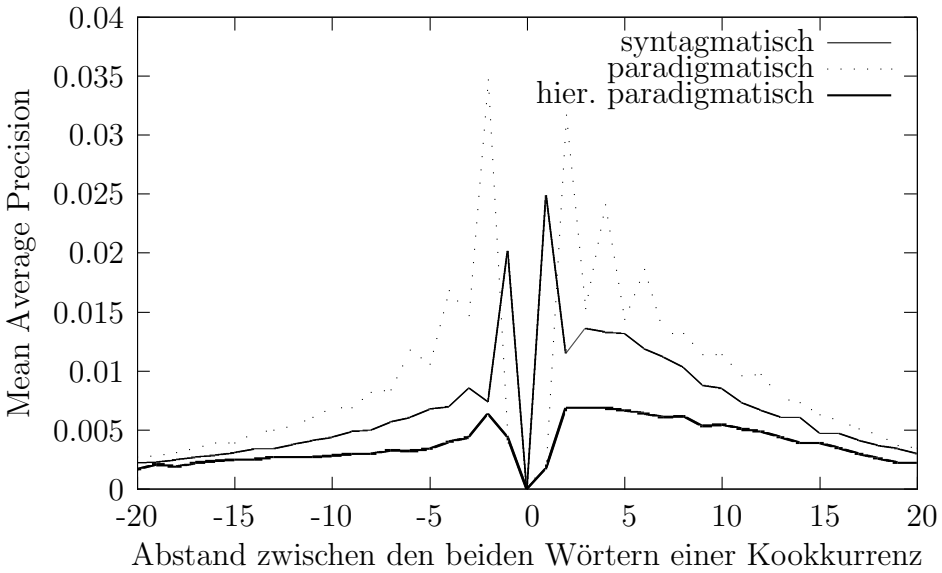

Abbildung 4.15: Verteilung der *syntagmatischen*, *paradigmatischen* und *hierarchisch-paradigmatischen* Relation

Andererseits ist ersichtlich, dass die *syntagmatische* Relation gegenüber der *paradigmatischen* Relation in den Abständen -1 und 1 eine zehnfach höhere *Mean Average Precision* hat (vgl. Anhang A, Tabelle A.1). Dies läßt sich für den Abstand 1 beispielsweise auf *Adjektiv-Substantiv*-Konstruktionen (vgl. *A N typische Eigenschaft* aus Tabelle A.1) zurückführen. Während für solche Konstruktionen im Abstand 1 *2184* Kookkurrenzen in der Annotationsdatenbank bekannt sind, können bei einem Abstand von 2 nur noch *417* und bei einem Abstand von 5 nur noch *290* Kookkurrenzen in der Datenbank gefunden werden.

Die *paradigmatische* Relation hingegen hat in den Abständen -1 und 1 eine sehr kleine *Mean Average Precision*, die allerdings bereits in den Abständen -2 und 2 ihre höchsten Peaks erreicht. Ausschlaggebend für diesen Anstieg sind vor allem die *Kohyponyme* und *Gegenteil*-Beziehungen zwischen Substantiven. Besonders die *Kohyponyme* zwischen Substantiven sind der Grund dafür, dass die *paradigmatische* Relation auch über größere Abstände hinweg die Relation mit der höchsten *Mean Average Precision* ist.

Eine weitere Beobachtung aus Abbildung 4.15 kann für die *paradigmatische* Relation gemacht werden. In kleineren Abständen besitzt der Graph deutlich mehr Peaks als die beiden anderen Relationen. Diese Beobachtung kann auf Aufzählungen zurückgeführt werden. Dabei entsprechen die Peaks

den Wörtern und die „Täler“ den Kommas aus der Aufzählung.

Als letzte Relation in diesem Kapitel sollen die *Derivate* betrachtet werden. In Abbildung 4.16 ist das Spektrum der Relation *Derivate* dargestellt. Entgegen der *syntagmatischen*, *paradigmatischen* und *hierarchisch-paradigmatischen* Relation nimmt die *Mean Average Precision* für die Relation *Derivate* bis zu den Abständen -7 und 7 zu und bleibt bis zu einem Abstand von etwa -18 und 17 auf einem hohen Niveau. Erst danach fällt die *Mean Average Precision* wieder leicht ab. Es sei allerdings daraufhingewiesen, dass die Relation *Derivate* die kleinste *Mean Average Precision* besitzt und damit die schwächste der hier vorgestellten Relationen ist.

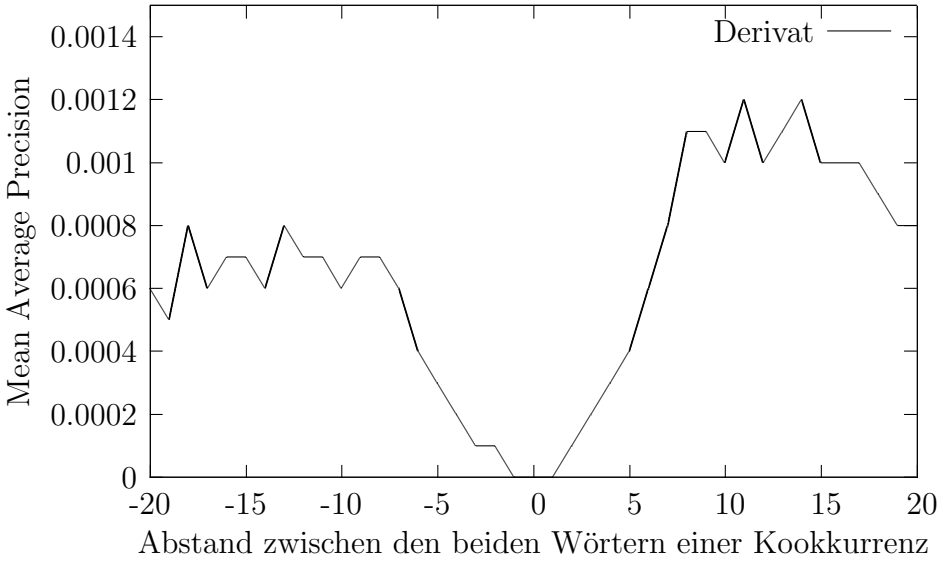

Abbildung 4.16: Verteilung von Derivaten

4.2 Signifikanzen in Abhängigkeit vom Abstand

Bei den bisherigen Betrachtungen konnten zwei Probleme identifziert werden. Einerseits sind Kookkurrenzen zwischen seltenen Wörter, die nur einmal beobachtet werden können, bereits signifikant. Der Grund dafür liegt in der stochastischen Unabängigkeit. Andererseits konnte im Abschnitt 3.2 gezeigt werden, dass auch zwischen hochfrequenten Wörtern Kookkurrenzen

als signifikant angesehen werden, die aber selbst bei einem zufälligen Korpus signifikant gewesen wären.

An dieser Stelle soll die Automatische Sprachverarbeitung einmal kurz verlassen werden. Unser Sonnensystem besteht aus einer Sonne sowie verschiedenen Planeten und deren Monde. Zwischen diesen Planeten herrschen Anziehungskräfte, die auch als Gravitationskräfte (siehe Formel 4.6) bezeichnet werden. Die Gravitationskraft hängt dabei von den Massen der zweier Planeten und deren Abstand ab. So können zwei massereiche Planeten (in der ASV beispielsweise zwei Substantive) deutlich weiter von einander entfernt sein als zwei massearme Planeten. K steht dabei für eine Gravitationskonstante, die nachfolgend nicht weiter von Bedeutung ist.

$$F_G(m_1, m_2, r) = -K\frac{m_1 * m_2}{r^2} \tag{4.6}$$

Bezogen auf die bisherigen Signifikanzmaße würde dies bedeuten, dass lediglich die beiden „Massen“[2] von Interesse sind. Der Vergleich zwischen Gravitation und Signifikanz ist deswegen sinnvoll, da beide messen, wie stark eine „Anziehung“ zwischen zwei Objekten ist.

Wird das Signifikanzmaß als das Produkt der beiden Massen aus Formel 4.6 aufgefaßt, dann kann die Formel 4.7 abgeleitet werden. Dies ist natürlich keine wissenschaftliche Ableitung. Vielmehr soll es eine Motivation bzw. eine Gedankenskizze sein, um Signifikanzen von Kookkurrrenzen auf Basis deren Abstände zu berechnen.

$$sig_{LMI,dist} = \frac{sig_{LMI}}{r^2} \tag{4.7}$$

Da es in der Automatischen Sprachverarbeitung verschiedene Problemstellungen, wie Synonyme oder Kohyponyme, gibt, ist es nicht sinnvoll einen größeren Abstand mit r^2 „zu bestrafen“, da aufgrund der Satzstruktur ggf. semantische Zusammenhänge erst in größeren Abständen auftreten (vgl. Kapitel 4.1.3). Deswegen soll r^2 durch eine Abstandsfunktion $f(r)$ mit dem Abstand r ersetzt werden.

$$sig_{LMI,dist} = \frac{sig_{LMI}}{f(r)} \tag{4.8}$$

Im Rahmen dieser Arbeit konnte gezeigt werden, dass das Spektrum für verschiedene Kookkurrenzen gleicher Art ähnlich ist. Auf Basis dieser Informationen lassen sich für $f(r)$ verschiedene Filterfunktionen konstruieren, die in einem bestimmten Abstandsbereich Kookkurrenzen durchlassen und

[2]Dies könnte im trivialsten Fall die Frequenz eines Wortes sein. Idealerweise sollte hier eher eine semantische „Masse“ darunter verstanden werden.

in den entsprechend anderen Abständen die Kookkurrenzen herausgefiltert werden. Im Sinne des *significance by semantic relation* aus Kapitel 3.3 kann beispielsweise eine Kookkurrenz *(abciger,Fußball)* als signifikant erachtet werden, wenn diese Kookkurrenz mit einem Abstand von 1 auftritt und *abciger* ein bisher neues Wort ist. *iger* deutet auf ein Adjektiv hin und aus Abbildung 4.15 ist bekannt, dass bei diesem Abstand vorrangig *syntagmatische Relationen* wie *A N typische Eigenschaft* beobachtet werden können.

Eine weitere interessante Abstandsfunktion ist die *Varianz* der Abstände einer Kookkurrenz. Es konnte bereits gezeigt werden, dass für verschiedene semantische Relationen unterschiedlich große Abstände zwischen den Wörtern gemessen werden konnten. Um aber alle Kookkurrenzen zu exportieren, die in einer beliebigen semantischen Relation stehen, sollte ein allgemeiner Formalismus benutzt werden, anstelle mit jeweiligen Abstandsfunktionen für jede semantische Relation die Signifikanz separat zu berechnen. Dazu kann die Varianz benutzt werden. Dabei bezieht sich die Varianz auf die unterschiedlichen Abstände, in welchen eine Kookkurrenz aufgetreten ist. Tritt eine Kookkurrenz mit den Abständen 3, 4 und 5 auf, so wird die Varianz sehr klein sein. Kookkurrenzen, die jedoch mit Abständen von 1-20 nahezu komplett aufgetreten sind, haben eine sehr hohe Varianz.

Das Problem beim Benutzen der Varianz ist jedoch, dass die meisten Kookkurrenzen ein sehr großes Spektrum haben, so dass teilweise die hohen Signifikanzen durch eine sehr hohe Varianz auf Werte unter 1 herunter gerechnet werden. Dann ist das Verhalten ähnlich der *Mutual Information*.

Nach einigen Tests scheint es daher sinnvoller zu sein, die Anzahl der Abstände zu zählen, für die die Beobachtung einer Kookkurrenz über der Erwartung und in einer zweiten Zahl die Anzahl der Abstände, für die die Beobachtung unter der Erwartung geblieben ist. Daraus ergeben sich dann beliebige Funktionen $f(r)$ wie beispielsweise der Quotient beider Zahlen.

4.3 Zusammenfassung

In diesem Kapitel konnte gezeigt, werden, dass eine *Spektralanalyse von Kookkurrenzen* einerseits machbar und andererseits auch wertvolle Ergebnisse liefert. Bei einer Spektralanalyse wie beispielsweise dem Spektrum der Distanz wird eine Kookkurrenz zerlegt. So werden aus rund 1,5 Mrd. Kookkurrenzen vor der Spektralanalyse etwa 3,4 Mrd. nach einer Zerlegung.

Als Ergebnis kann festgehalten werden, dass eine Kombinationen von Kookkurrenzen mit Strukturinformationen aus dem Satz spezialisiertere Ergebnisse liefert. Bei der herkömmlichen Kookkurrenzanalyse wird das Auftreten einer Kookkurrenz in einem vorher definierten Fenster gezählt. Dabei

ist die Wahl der Fenstergröße entscheidend für das Ergebnis.

Die *Spektralanalyse von Kookkurrenzen* hat dies bestätigt. Vielmehr bietet sie aber die Möglichkeit an, dass bestimmte Frequenzteile als Rauschen herausgefiltert werden können. Dadurch besteht die Möglichkeit Kookkurrenzen in den Abständen zu untersuchen, die einer gegebenen Forschungsfrage entsprechen. So konnte beispielsweise gezeigt werden, dass *Adjektiv-Substantiv*-Verbindungen nur in einem sehr kleinen Fenster beobachtet werden können. Andererseits zeigt sich auch, dass Kookkurrenzen zwischen *Substantiven* auch in größeren Fenstern beobachtet werden können.

An dieser Stelle wird der Nutzen der Spektralanalyse sichtbar. Während bei der Berechnung von Satzkookkurrenzen bisher alle möglichen Wortkombinationen gezählt werden, zeigt sich deutlich, dass dies gar nicht sinnvoll ist. Kookkurrenzen, deren Wörter keine Substantive sind und die im Satz eine hinreichend großen Abstand besitzen, brauchen demnach gar nicht gezählt werden. Mit Ausnahme der *Derivate*.

In Kapitel 3 sind einige Signifikanzmaße vorgestellt worden. Diese Maße können jedoch nicht mit Abständen umgehen. Im Rahmen dieses Kapitels konnte gezeigt werden, dass eine Kombination dieser Maße mit einer Abstandsfunktion eine Verbesserung beim Extrahieren von signifikanten Kookkurrenzen darstellt. Dabei steht nicht mehr nur die statistische Auffälligkeit im Vordergrund, sondern auch die Syntax eines Satzes.

Kapitel 5

Zusammenfassung

Diese Arbeit besteht aus drei relevanten Teilen. In Kapitel 2 wird gezeigt, wie das Zählen von Kookkurrenzen effektiv gestaltet werden kann. Probleme hierbei stellt die *Zipfverteilung* der Kookkurrenzen dar, da die Hashfunktion diese starke Ungleichverteilung ausgleichen muß. Als ein Maß für diesen Ausgleich wird in Kapitel 2 der *Avalanche-Effekt* definiert und diverse Hashfunktionen bezüglich zipfverteilter Daten untersucht. Dabei haben vor allem die Algorithmen, die aus der Kryptologie stammen, sehr gute Ergebnisse geliefert. Andererseits hat sich gezeigt, dass der *CRC32*-Algorithmus sehr schnell und somit die Wahl auf ihn gefallen ist.

Es wird weiterhin gezeigt, dass das Benutzen eines speziellen *Modulo*-Operators wesentlich effektiver ist als der Standardalgorithmus der Programmiersprache Java. Dadurch konnte der Durchsatz in Sätzen/Minuten um etwa 20% erhöht werden.

Abschließend wird gezeigt, dass sich bei einem geeigneten Verhältnis von perfektem und offenem Hashing einerseits die Verarbeitungsgeschwindigkeit verbessert und andererseits die Anzahl der im Speicher gezählten Kookkurrenzen erhöht wird.

Im nächsten relevanten Kapitel werden verschiedene Signifikanzmaße auf deren signifikantesten und unsignifikantesten Kookkurrenzen untersucht. Dabei wird festgestellt, dass die *Local Mutual Information*, das *Log-Likelihood*-Ratio und das *Poisson*-Maß für die signifikant häufigen Kookkurrenzen das Gleiche berechnen. Bei den signifikant seltenen Kookkurrenzen hingegen unterscheiden sich die Ergebnisse des *Log-Likelihood*-Ratio von den beiden anderen Maßen deutlich.

Im letzten Kapitel wird gezeigt, dass das Berechnen von spektral zerlegten Kookkurrenzen aus mehreren Gründen sinnvoll ist. Einerseits erübrigt sich dabei die Diskussion einer Distinktion der Wörter innerhalb eines Satzfensters, da durch das spektrale Zählen mögliche Probleme vollständig

vermieden werden können. Andererseits wird gezeigt, dass sich bestimmte semantische Relationen in verschiedenen Abständen bevorzugt wiederfinden. Beispielsweise sind im Abstand -1 und 1 die syntagmatischen Relationen die häufigsten Beziehungen zwischen zwei Wörtern. Bei Abständen von kleiner -2 und größer 2 ist die paradigmatische Relation dominierend. Weiterhin wird gezeigt, dass die Relation der Derivate mit zunehmendem Abstand stärker ausgeprägt ist und erst bei Abständen von fast 20 wieder abschwächt.

Die im Rahmen dieser Arbeit vorgestellten Ergebnisse sind nur ein Anfang in der spektralen Kookkurrenzanalyse. Einerseits geben die vorgestellten Spektren nur einen ersten Eindruck von dem, wozu sie genutzt werden koennen. Andererseits ergeben sich auch neue Aufgaben wie die spektrale Kookkurrenzanalyse eines *Part-Of-Speech* getaggten Korpus, auf welchem Kookkurrenzen zwischen POS-Tags berechnet werden können. Hierbei stehen schließlich die Spektren verschiedener Kookkurrenzen zwischen POS-Tags im Mittelpunkt. Gegebenenfalls kann daraus sogar ein neuer Ansatz des Part-Of-Speech-Taggings abgeleitet werden. Auf jeden Fall ist jedoch zu erwarten, dass die Ergebnisse zur Verbesserung bestehender POS-Tagger bzw. für die automatische Generierung von natürlich-sprachlichen Texten genutzt werden können.

Literaturverzeichnis

[Bar99] Hans-Jochen Bartsch. *Taschenbuch Mathematischer Formeln*. Carl Hanser Verlag, Leipzig, 1999.

[Bau94] Friedrich L. Bauer. *Kryptologie - Methoden und Maximen*. Springer Verlag, Berlin, second edition, 1994.

[BB04] Marco Baroni and Sabrina Bisi. Using cooccurrence statistics and the web to discover synonyms in technical language, 2004.

[BH05] Stefan Bordag and Gerhard Heyer. *A Structuralist Framework for Quantitative Linguistics*. Studies in Fuzziness and Soft Computing. Springer, Berlin/Heidelberg/New York, 2005.

[Bie06a] Chris Biemann. Implementierung einer Kookkurrenzmaschine: Howto − Reimplementierung ConceptComposer, 2006.

[Bie06b] Christian Biemann. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL-06 Student Research Workshop 2006, Sydney, Australia*, 2006.

[Boh91] Peter Bohley. *Statistik - Einführendes Lehrbuch für Wirtschafts- und Sozialwissenschaftler*. R. Oldenbourg Verlag, München Wien, fourth edition, 1991.

[Bor06] Stefan Bordag. Two-step approach to unsupervised morpheme segmentation. In *Proceedings of the Unsupervised segmentation of words into morphemes - Challenge 2005. Venice, Italy*, 2006.

[CG95] Kenneth Ward Church and William A. Gale. Poisson mixtures. *Journal of Natural Language Engineering*, 1(2):163–190, 1995.

[CGHH91] Kenneth Ward Church, William A. Gale, Patrick Hanks, and Donald Hindle. Using statistics in lexical analysis. In Uri Zernik,

editor, *Lexical Acquisition: Exploiting On-Line Resources to Build up a Lexicon*, pages 115–164, Hillsdale, NJ, 1991. Lawrence Erlbaum.

[CH90] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, 1990.

[Dic45] Lee R. Dice. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26(3):297–302, 1945.

[dS01] Ferdinand de Saussure. *Grundfragen der allgemeinen Sprachwissenschaft*. de Gruyter, 3rd edition, 2001. C. Bally and A. Sechehaye (eds.).

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, 1993.

[Eve04] Stefan Evert. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD thesis, Universität Stuttgart, August 2004.

[Fir57] J.R. Firth. *A synopsis of linguistic theory 1930-1955.* Oxford: Philological Society., reprinted in f. r. palmer (ed), selected papers of j.r. firth 1952-1959, london: longman, 1968 edition, 1957.

[FS01] Ramon Ferrer-i-Cancho and Ricard V. Sole. Two regimes in the frequency of words and the origins of complex lexicons: Zipf's law revisited. *Journal of Quantitative Linguistics*, 8(3):165–173, 2001.

[FS03] Ramon Ferrer-i-Cancho and Ricard V. Sole. Least effort and the origins of scaling in human language. *PNAS*, 100:788–791, 2003.

[GB] Zhong Gu and Daniel Berleant. Hash table sizes for storing n-grams for text processing.

[Har51] Zellig S. Harris. *Structural Linguistics*. University of Chicago Press, Chicago, 1951.

[HQW06] Gerhard Heyer, Uwe Quasthoff, and Thomas Wittig. *Text Mining: Wissensrohstoff Text - Konzepte, Algorithmen, Ergebnisse*. W3L-Verlag, Herdecke, Bochum, 2006.

[HR01] Theo Härder and Erhard Rahm. *Datenbankensysteme - Konzepte und Techniken der Implementierung*. Springer Verlag, Berlin, second edition, 2001.

[Knu98] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, second edition, 1998.

[Lan04] Stefan Langer. Statistische Methoden in der Sprachverarbeitung, 2004.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

[MH02] Jochen Müsseler and Wolfgang Prinz (Hrsg.). *Allgemeine Psychologie*. Spektrum Akademischer Verlag, Heidelberg Berlin, first edition, 2002.

[New05] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323, 2005.

[OW96] T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen*. Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford, third edition, 1996.

[Pag99] Rasmus Pagh. Hash and displace: Efficient evaluation of minimal perfect hash functions. In *Workshop on Algorithms and Data Structures*, pages 49–54, 1999.

[Pag02] Rasmus Pagh. Hashing, randomness and dictionaries, 2002.

[Pec05] Pavel Pecina. An Extensive Empirical Study of Collocation Extraction Methods. In *Proceedings of the ACL Student Research Workshop*, pages 13–18, Ann Arbor, Michigan, USA, June 25-30, 2005. Association for Computational Linguistics.

[PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.

[Qua98] Uwe Quasthoff. Projekt: Der Deutsche Wortschatz. In Gerhard Heyer and Christian Wolff, editors, *Tagungsband zur GLDV-Tagung*, pages 93–99, Leipzig, March 1998. Deutscher Universitätsverlag.

[QW02] Uwe Quasthoff and Christian Wolff. The poisson collocation measure and its applications. In *Second International Workshop on Computational Approaches to Collocations*, 2002.

[Rap02] Reinhard Rapp. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[Sal89] Gerard Salton. *Automatic Text Processing - The Transformation, Analysis and Retrieval of Informationby Computer.* Addison-Wesley Publishing Company, Menlo Park, California, first edition, 1989.

[SBW05] F.H. Witschel S. Bordag and T. Wittig. Evaluation of lexical acquisition algorithms. In Bonn, Peter-Lang-Verlag, Frankfurt am Main, editor, *Sprache, Sprechen und Computer/Computer Studies in Language and Speech und Proceedings of GLDV-Frühjahrstagung*, 2005.

[Sch96] Bruce Schneier. *Applied Cryptography - Protocols, Algorithms and Source in C.* John Wiley & Sons, Inc, New York, second edition, 1996.

[Sha48] Claude E. Shannon. A mathematical theory of communications, 1948.

[sP02] Anna Östlin and Rasmus Pagh. Simulating uniform hashing in constant time and optimal space, 2002.

[Tan58] T.T. Tanimoto. An element mathematical theory of classification. Technical report, I.B.M. Research, New York, NY USA, November 1958.

[Tes92] Maria Tesitelova. *Quantitative Linguistics.* John Benjamins Publishing Company, Amsterdam, Philadelphia, 1992.

[Top73] Flemming Topsoe. *Informationstheorie.* B. G. Teubner Studienbücher, Stuttgart, 1973.

[Völ91] Horst Völz. *Grundlagen der Information.* Akademie Verlag GmbH, Berlin, 1991.

[Wik05] Wikipedia. Wikipedia, the free encyclopedia. Web site: http://wikipedia.org, 2005.

[WR93] M. Wettler and R. Rapp. Computation of word associations based on the co-occurrences of words in large corpora, 1993.

[ZHW01] Justin Zobel, Steffen Heinz, and Hugh E. Williams. In-memory hash tables for accumulating text vocabularies. *Information Processing Letters*, 80(6):271–277, 2001.

[Zim92] Philip G. Zimbardo. *Psychologie*. Springer Verlag, Berlin, fourth edition, 1992.

[Zip49] George Kingsley Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.

Anhang A

Spektrale Zerlegungen von semantischen Relationen

Dieser Anhang stellt die Daten verschiedener semantischer Relationen nach einer *spektralen Kookkurrenzanalyse* zur Verfügung. Dabei wird in insgesamt 20 Tabellen je der positive sowie der negative Abstand in einer Tabelle dargestellt. Insgesamt werden die Ergebnisse aus den Abständen -20 bis 20 für ausgewählte Relationen abgebildet.

Die Ergebnisse basieren auf dem *de*-Korpus aus dem *Wortschatz*-Projekt (vgl. [Qua98]). Dieser Korpus besteht aus 35,7 Millionen deutschen Sätzen. Die Kookkurrenzen wurden mit der höchstmöglichen Auflösung der spektralen Zerlegung gezählt. Dies bedeutet, dass die Kookkurrenzen in jedem Abstand separat gezählt worden sind.

Relation	Abstand=-1	Abstand=1
A A Gegenteil	0,0001 (*5*)	0,0000 (*5*)
A A Kohyponym	0,0008 (*331*)	0,0005 (*348*)
A A Synonym	0,0001 (*11*)	0,0000 (*13*)
A N typische Eigenschaft	0,0007 (*148*)	0,0114 (*2184*)
A V typische Eigenschaft	0,0002 (*64*)	0,0010 (*217*)
N N Beruf zu	0,0000 (*2*)	0,0001 (*19*)
N N Chef von	0,0000 (*3*)	0,0000 (*6*)
N N Eigenname zu	0,0030 (*295*)	0,0005 (*87*)
N N Gegenteil	0,0001 (*5*)	0,0000 (*5*)
N N Grund von	0,0000 (*1*)	0,0000 (*4*)
N N hat Aggregat	0,0000 (*3*)	0,0000 (*1*)
N N hat Beruf	0,0003 (*19*)	0,0000 (*2*)
N N hat Chef	0,0000 (*6*)	0,0000 (*3*)
N N hat Teil/hat Material	0,0000 (*8*)	0,0001 (*12*)
N N hat typischen Ort	0,0002 (*39*)	0,0004 (*42*)
N N Kohyponym	0,0023 (*876*)	0,0014 (*914*)
N N Oberbegriff	0,0000 (*22*)	0,0013 (*342*)
N N Synonym	0,0007 (*41*)	0,0003 (*41*)
N N Teil von/Material von	0,0002 (*12*)	0,0001 (*11*)
N N typischer Ort für	0,0003 (*42*)	0,0002 (*39*)
N N Unterbegriff	0,0036 (*339*)	0,0002 (*22*)
N V typisches Objekt/Instrument von	0,0014 (*317*)	0,0102 (*2408*)
N V typisches Subjekt von	0,0008 (*111*)	0,0029 (*705*)
V A hat typische Eigenschaft	0,0014 (*197*)	0,0004 (*67*)
V N hat typisches Objekt/Instrument	0,0054 (*2253*)	0,0007 (*324*)
V N typische Tätigkeit	0,0025 (*656*)	0,0002 (*113*)
V N typische Tätigkeit für Ort	0,0006 (*103*)	0,0000 (*3*)
V V Gegenteil	0,0000 (*1*)	0,0000 (*1*)
V V Kohyponym	0,0001 (*41*)	0,0001 (*76*)
V V Synonym	0,0000 (*0*)	0,0000 (*1*)
syntagmatic	0,0202 (*5993*)	0,0249 (*6339*)
paradigmatic	0,0039 (*1362*)	0,0024 (*1456*)
hierarch.paradigmatic	0,0044 (*571*)	0,0018 (*586*)
derivation	0,0000 (*8*)	0,0000 (*9*)
other	0,0041 (*416*)	0,0020 (*420*)
total	0,0261 (*8138*)	0,0220 (*8589*)

Tabelle A.1: Semant. Relationen im Abstand -1 und 1

Relation	Abstand=-2	Abstand=2
A A Gegenteil	0,0017 (*182*)	0,0011 (*182*)
A A Kohyponym	0,0033 (*1027*)	0,0024 (*1048*)
A A Synonym	0,0010 (*155*)	0,0007 (*155*)
A N typische Eigenschaft	0,0005 (*153*)	0,0016 (*417*)
A V typische Eigenschaft	0,0003 (*77*)	0,0004 (*95*)
N N Beruf zu	0,0002 (*33*)	0,0003 (*63*)
N N Chef von	0,0002 (*51*)	0,0005 (*71*)
N N Eigenname zu	0,0010 (*222*)	0,0003 (*83*)
N N Gegenteil	0,0029 (*312*)	0,0025 (*319*)
N N Grund von	0,0007 (*93*)	0,0005 (*99*)
N N hat Aggregat	0,0003 (*52*)	0,0003 (*44*)
N N hat Beruf	0,0003 (*61*)	0,0002 (*33*)
N N hat Chef	0,0005 (*69*)	0,0003 (*55*)
N N hat Teil/hat Material	0,0016 (*497*)	0,0012 (*228*)
N N hat typischen Ort	0,0013 (*240*)	0,0036 (*567*)
N N Kohyponym	0,0246 (*12510*)	0,0252 (*12743*)
N N Oberbegriff	0,0012 (*377*)	0,0015 (*490*)
N N Synonym	0,0039 (*544*)	0,0031 (*550*)
N N Teil von/Material von	0,0011 (*224*)	0,0034 (*521*)
N N typischer Ort für	0,0014 (*515*)	0,0011 (*249*)
N N Unterbegriff	0,0028 (*479*)	0,0021 (*386*)
N V typisches Objekt/Instrument von	0,0015 (*654*)	0,0032 (*888*)
N V typisches Subjekt von	0,0010 (*374*)	0,0006 (*157*)
V A hat typische Eigenschaft	0,0003 (*90*)	0,0005 (*78*)
V N hat typisches Objekt/Instrument	0,0017 (*865*)	0,0014 (*675*)
V N typische Tätigkeit	0,0005 (*152*)	0,0013 (*386*)
V N typische Tätigkeit für Ort	0,0002 (*44*)	0,0001 (*14*)
V V Gegenteil	0,0006 (*72*)	0,0005 (*72*)
V V Kohyponym	0,0019 (*389*)	0,0015 (*424*)
V V Synonym	0,0004 (*47*)	0,0003 (*47*)
syntagmatic	0,0074 (*3562*)	0,0115 (*3708*)
paradigmatic	0,0347 (*15166*)	0,0317 (*15460*)
hierarch.paradigmatic	0,0064 (*1902*)	0,0069 (*1953*)
deriv	0,0001 (*17*)	0,0001 (*17*)
other	0,0038 (*803*)	0,0040 (*822*)
total	0,0418 (*20932*)	0,0410 (*21422*)

Tabelle A.2: Semant. Relationen im Abstand -2 und 2

Relation	Abstand=-3	Abstand=3
A A Gegenteil	0,0013 (*186*)	0,0012 (*186*)
A A Kohyponym	0,0015 (*509*)	0,0012 (*534*)
A A Synonym	0,0003 (*46*)	0,0002 (*48*)
A N typische Eigenschaft	0,0009 (*246*)	0,0020 (*482*)
A V typische Eigenschaft	0,0003 (*92*)	0,0002 (*62*)
N N Beruf zu	0,0002 (*65*)	0,0004 (*92*)
N N Chef von	0,0002 (*38*)	0,0005 (*92*)
N N Eigenname zu	0,0007 (*179*)	0,0008 (*175*)
N N Gegenteil	0,0020 (*295*)	0,0023 (*303*)
N N Grund von	0,0004 (*65*)	0,0005 (*88*)
N N hat Aggregat	0,0003 (*56*)	0,0004 (*63*)
N N hat Beruf	0,0004 (*86*)	0,0004 (*71*)
N N hat Chef	0,0004 (*88*)	0,0002 (*41*)
N N hat Teil/hat Material	0,0014 (*465*)	0,0011 (*282*)
N N hat typischen Ort	0,0019 (*416*)	0,0049 (*877*)
N N Kohyponym	0,0101 (*6565*)	0,0116 (*6950*)
N N Oberbegriff	0,0011 (*530*)	0,0012 (*353*)
N N Synonym	0,0016 (*277*)	0,0014 (*292*)
N N Teil von/Material von	0,0013 (*270*)	0,0032 (*519*)
N N typischer Ort für	0,0018 (*775*)	0,0014 (*427*)
N N Unterbegriff	0,0017 (*334*)	0,0032 (*593*)
N V typisches Objekt/Instrument von	0,0017 (*813*)	0,0035 (*997*)
N V typisches Subjekt von	0,0007 (*284*)	0,0011 (*281*)
V A hat typische Eigenschaft	0,0002 (*60*)	0,0006 (*93*)
V N hat typisches Objekt/Instrument	0,0017 (*980*)	0,0020 (*883*)
V N typische Tätigkeit	0,0009 (*275*)	0,0008 (*319*)
V N typische Tätigkeit für Ort	0,0001 (*38*)	0,0003 (*45*)
V V Gegenteil	0,0003 (*56*)	0,0004 (*56*)
V V Kohyponym	0,0006 (*158*)	0,0005 (*164*)
V V Synonym	0,0001 (*18*)	0,0001 (*18*)
syntagmatic	0,0086 (*4406*)	0,0136 (*4696*)
paradigmatic	0,0147 (*8068*)	0,0153 (*8509*)
hierarch.paradigmatic	0,0044 (*1911*)	0,0069 (*2065*)
deriv	0,0001 (*29*)	0,0002 (*30*)
other	0,0034 (*875*)	0,0040 (*924*)
total	0,0221 (*14604*)	0,0276 (*15485*)

Tabelle A.3: Semant. Relationen im Abstand -3 und 3

Relation	Abstand=-4	Abstand=4
A A Gegenteil	0,0014 (*242*)	0,0017 (*243*)
A A Kohyponym	0,0015 (*719*)	0,0017 (*742*)
A A Synonym	0,0003 (*63*)	0,0003 (*64*)
A N typische Eigenschaft	0,0007 (*249*)	0,0010 (*308*)
A V typische Eigenschaft	0,0002 (*78*)	0,0004 (*76*)
N N Beruf zu	0,0002 (*53*)	0,0004 (*89*)
N N Chef von	0,0002 (*59*)	0,0005 (*92*)
N N Eigenname zu	0,0006 (*153*)	0,0011 (*201*)
N N Gegenteil	0,0019 (*297*)	0,0023 (*302*)
N N Grund von	0,0003 (*66*)	0,0006 (*96*)
N N hat Aggregat	0,0003 (*61*)	0,0003 (*56*)
N N hat Beruf	0,0003 (*86*)	0,0004 (*56*)
N N hat Chef	0,0003 (*87*)	0,0004 (*61*)
N N hat Teil/hat Material	0,0010 (*377*)	0,0014 (*348*)
N N hat typischen Ort	0,0018 (*462*)	0,0033 (*638*)
N N Kohyponym	0,0124 (*10202*)	0,0194 (*10545*)
N N Oberbegriff	0,0010 (*550*)	0,0014 (*392*)
N N Synonym	0,0013 (*290*)	0,0014 (*301*)
N N Teil von/Material von	0,0015 (*336*)	0,0023 (*412*)
N N typischer Ort für	0,0011 (*572*)	0,0019 (*480*)
N N Unterbegriff	0,0016 (*384*)	0,0034 (*588*)
N V typisches Objekt/Instrument von	0,0014 (*797*)	0,0053 (*1317*)
N V typisches Subjekt von	0,0006 (*250*)	0,0014 (*339*)
V A hat typische Eigenschaft	0,0003 (*76*)	0,0005 (*80*)
V N hat typisches Objekt/Instrument	0,0018 (*1292*)	0,0019 (*839*)
V N typische Tätigkeit	0,0009 (*326*)	0,0008 (*277*)
V N typische Tätigkeit für Ort	0,0002 (*44*)	0,0003 (*52*)
V V Gegenteil	0,0004 (*80*)	0,0004 (*80*)
V V Kohyponym	0,0007 (*241*)	0,0007 (*248*)
V V Synonym	0,0001 (*29*)	0,0001 (*29*)
syntagmatic	0,0070 (*4408*)	0,0133 (*4626*)
paradigmatic	0,0168 (*12142*)	0,0241 (*12530*)
hierarch.paradigmatic	0,0040 (*1939*)	0,0069 (*2048*)
deriv	0,0002 (*49*)	0,0003 (*51*)
other	0,0029 (*891*)	0,0044 (*934*)
total	0,0219 (*18704*)	0,0358 (*19417*)

Tabelle A.4: Semant. Relationen im Abstand -4 und 4

Relation	Abstand=-5	Abstand=5
A A Gegenteil	0,0011 (*202*)	0,0014 (*202*)
A A Kohyponym	0,0009 (*500*)	0,0010 (*522*)
A A Synonym	0,0002 (*48*)	0,0002 (*49*)
A N typische Eigenschaft	0,0008 (*270*)	0,0010 (*290*)
A V typische Eigenschaft	0,0002 (*75*)	0,0003 (*53*)
N N Beruf zu	0,0003 (*73*)	0,0005 (*120*)
N N Chef von	0,0001 (*51*)	0,0006 (*112*)
N N Eigenname zu	0,0005 (*139*)	0,0012 (*201*)
N N Gegenteil	0,0018 (*287*)	0,0026 (*294*)
N N Grund von	0,0002 (*67*)	0,0007 (*106*)
N N hat Aggregat	0,0002 (*50*)	0,0004 (*61*)
N N hat Beruf	0,0004 (*114*)	0,0005 (*77*)
N N hat Chef	0,0003 (*103*)	0,0003 (*55*)
N N hat Teil/hat Material	0,0008 (*373*)	0,0016 (*384*)
N N hat typischen Ort	0,0017 (*521*)	0,0040 (*656*)
N N Kohyponym	0,0072 (*6294*)	0,0105 (*6613*)
N N Oberbegriff	0,0009 (*522*)	0,0010 (*328*)
N N Synonym	0,0012 (*295*)	0,0015 (*310*)
N N Teil von/Material von	0,0017 (*379*)	0,0026 (*399*)
N N typischer Ort für	0,0011 (*610*)	0,0019 (*536*)
N N Unterbegriff	0,0013 (*322*)	0,0031 (*553*)
N V typisches Objekt/Instrument von	0,0014 (*711*)	0,0051 (*1333*)
N V typisches Subjekt von	0,0005 (*245*)	0,0013 (*358*)
V A hat typische Eigenschaft	0,0002 (*50*)	0,0005 (*76*)
V N hat typisches Objekt/Instrument	0,0017 (*1289*)	0,0016 (*753*)
V N typische Tätigkeit	0,0009 (*350*)	0,0009 (*268*)
V N typische Tätigkeit für Ort	0,0001 (*44*)	0,0003 (*48*)
V V Gegenteil	0,0003 (*83*)	0,0006 (*84*)
V V Kohyponym	0,0005 (*172*)	0,0005 (*176*)
V V Synonym	0,0002 (*32*)	0,0001 (*32*)
syntagmatic	0,0068 (*4420*)	0,0132 (*4621*)
paradigmatic	0,0106 (*7896*)	0,0144 (*8263*)
hierarch.paradigmatic	0,0034 (*1916*)	0,0067 (*2002*)
deriv	0,0003 (*80*)	0,0004 (*82*)
other	0,0029 (*943*)	0,0049 (*986*)
total	0,0158 (*14501*)	0,0269 (*15159*)

Tabelle A.5: Semant. Relationen im Abstand -5 und 5

Relation	Abstand=-6	Abstand=6
A A Gegenteil	0,0008 (*181*)	0,0013 (*181*)
A A Kohyponym	0,0010 (*594*)	0,0012 (*607*)
A A Synonym	0,0002 (*55*)	0,0003 (*55*)
A N typische Eigenschaft	0,0007 (*263*)	0,0009 (*249*)
A V typische Eigenschaft	0,0002 (*71*)	0,0002 (*52*)
N N Beruf zu	0,0003 (*70*)	0,0006 (*120*)
N N Chef von	0,0003 (*62*)	0,0006 (*109*)
N N Eigenname zu	0,0006 (*141*)	0,0008 (*161*)
N N Gegenteil	0,0016 (*288*)	0,0023 (*292*)
N N Grund von	0,0002 (*69*)	0,0006 (*101*)
N N hat Aggregat	0,0002 (*50*)	0,0004 (*57*)
N N hat Beruf	0,0005 (*113*)	0,0004 (*72*)
N N hat Chef	0,0004 (*103*)	0,0003 (*62*)
N N hat Teil/hat Material	0,0008 (*370*)	0,0017 (*388*)
N N hat typischen Ort	0,0018 (*494*)	0,0034 (*626*)
N N Kohyponym	0,0086 (*8158*)	0,0148 (*8455*)
N N Oberbegriff	0,0007 (*509*)	0,0010 (*382*)
N N Synonym	0,0013 (*342*)	0,0016 (*360*)
N N Teil von/Material von	0,0013 (*379*)	0,0028 (*399*)
N N typischer Ort für	0,0010 (*575*)	0,0021 (*512*)
N N Unterbegriff	0,0014 (*374*)	0,0027 (*534*)
N V typisches Objekt/Instrument von	0,0011 (*576*)	0,0044 (*1161*)
N V typisches Subjekt von	0,0005 (*200*)	0,0015 (*359*)
V A hat typische Eigenschaft	0,0002 (*50*)	0,0005 (*71*)
V N hat typisches Objekt/Instrument	0,0015 (*1133*)	0,0013 (*600*)
V N typische Tätigkeit	0,0009 (*350*)	0,0007 (*215*)
V N typische Tätigkeit für Ort	0,0002 (*47*)	0,0002 (*41*)
V V Gegenteil	0,0003 (*73*)	0,0005 (*73*)
V V Kohyponym	0,0005 (*169*)	0,0005 (*178*)
V V Synonym	0,0002 (*41*)	0,0002 (*41*)
syntagmatic	0,0061 (*3967*)	0,0119 (*4130*)
paradigmatic	0,0118 (*9876*)	0,0186 (*10212*)
hierarch.paradigmatic	0,0032 (*1910*)	0,0064 (*1995*)
deriv	0,0004 (*113*)	0,0006 (*116*)
other	0,0027 (*900*)	0,0041 (*934*)
total	0,0164 (*16020*)	0,0299 (*16599*)

Tabelle A.6: Semant. Relationen im Abstand -6 und 6

Relation	Abstand=-7	Abstand=7
A A Gegenteil	0,0009 (*184*)	0,0012 (*184*)
A A Kohyponym	0,0007 (*468*)	0,0010 (*480*)
A A Synonym	0,0002 (*49*)	0,0003 (*50*)
A N typische Eigenschaft	0,0006 (*259*)	0,0008 (*254*)
A V typische Eigenschaft	0,0001 (*58*)	0,0002 (*34*)
N N Beruf zu	0,0004 (*95*)	0,0005 (*126*)
N N Chef von	0,0001 (*52*)	0,0006 (*101*)
N N Eigenname zu	0,0006 (*149*)	0,0009 (*184*)
N N Gegenteil	0,0016 (*293*)	0,0024 (*294*)
N N Grund von	0,0001 (*53*)	0,0006 (*101*)
N N hat Aggregat	0,0003 (*63*)	0,0004 (*65*)
N N hat Beruf	0,0005 (*123*)	0,0005 (*97*)
N N hat Chef	0,0003 (*95*)	0,0003 (*54*)
N N hat Teil/hat Material	0,0007 (*349*)	0,0016 (*399*)
N N hat typischen Ort	0,0018 (*500*)	0,0038 (*659*)
N N Kohyponym	0,0058 (*5774*)	0,0095 (*6049*)
N N Oberbegriff	0,0006 (*528*)	0,0010 (*351*)
N N Synonym	0,0012 (*339*)	0,0015 (*352*)
N N Teil von/Material von	0,0018 (*389*)	0,0023 (*377*)
N N typischer Ort für	0,0013 (*614*)	0,0019 (*513*)
N N Unterbegriff	0,0014 (*337*)	0,0029 (*550*)
N V typisches Objekt/Instrument von	0,0008 (*464*)	0,0042 (*1077*)
N V typisches Subjekt von	0,0004 (*162*)	0,0015 (*374*)
V A hat typische Eigenschaft	0,0001 (*32*)	0,0005 (*61*)
V N hat typisches Objekt/Instrument	0,0013 (*1066*)	0,0010 (*491*)
V N typische Tätigkeit	0,0008 (*363*)	0,0006 (*173*)
V N typische Tätigkeit für Ort	0,0001 (*26*)	0,0002 (*41*)
V V Gegenteil	0,0002 (*64*)	0,0004 (*64*)
V V Kohyponym	0,0003 (*141*)	0,0004 (*148*)
V V Synonym	0,0001 (*45*)	0,0002 (*45*)
syntagmatic	0,0057 (*3754*)	0,0112 (*3888*)
paradigmatic	0,0084 (*7311*)	0,0132 (*7617*)
hierarch.paradigmatic	0,0033 (*1880*)	0,0061 (*1973*)
deriv	0,0006 (*156*)	0,0008 (*158*)
other	0,0029 (*949*)	0,0042 (*986*)
total	0,0134 (*13276*)	0,0245 (*13801*)

Tabelle A.7: Semant. Relationen im Abstand -7 und 7

Relation	Abstand=-8	Abstand=8
A A Gegenteil	0,0006 (*144*)	0,0010 (*144*)
A A Kohyponym	0,0007 (*448*)	0,0009 (*458*)
A A Synonym	0,0001 (*40*)	0,0002 (*42*)
A N typische Eigenschaft	0,0006 (*258*)	0,0008 (*203*)
A V typische Eigenschaft	0,0001 (*51*)	0,0003 (*43*)
N N Beruf zu	0,0002 (*69*)	0,0004 (*108*)
N N Chef von	0,0001 (*40*)	0,0007 (*109*)
N N Eigenname zu	0,0004 (*109*)	0,0008 (*170*)
N N Gegenteil	0,0014 (*254*)	0,0020 (*255*)
N N Grund von	0,0001 (*50*)	0,0006 (*87*)
N N hat Aggregat	0,0002 (*50*)	0,0004 (*56*)
N N hat Beruf	0,0005 (*106*)	0,0005 (*72*)
N N hat Chef	0,0003 (*108*)	0,0002 (*40*)
N N hat Teil/hat Material	0,0008 (*365*)	0,0017 (*390*)
N N hat typischen Ort	0,0017 (*493*)	0,0037 (*639*)
N N Kohyponym	0,0056 (*6111*)	0,0100 (*6338*)
N N Oberbegriff	0,0006 (*519*)	0,0010 (*381*)
N N Synonym	0,0012 (*314*)	0,0014 (*324*)
N N Teil von/Material von	0,0016 (*383*)	0,0027 (*391*)
N N typischer Ort für	0,0011 (*599*)	0,0018 (*504*)
N N Unterbegriff	0,0014 (*370*)	0,0031 (*535*)
N V typisches Objekt/Instrument von	0,0007 (*405*)	0,0035 (*986*)
N V typisches Subjekt von	0,0003 (*135*)	0,0014 (*365*)
V A hat typische Eigenschaft	0,0001 (*41*)	0,0004 (*51*)
V N hat typisches Objekt/Instrument	0,0010 (*971*)	0,0008 (*419*)
V N typische Tätigkeit	0,0009 (*354*)	0,0004 (*141*)
V N typische Tätigkeit für Ort	0,0001 (*41*)	0,0002 (*35*)
V V Gegenteil	0,0003 (*69*)	0,0005 (*70*)
V V Kohyponym	0,0002 (*129*)	0,0003 (*136*)
V V Synonym	0,0001 (*46*)	0,0002 (*47*)
syntagmatic	0,0050 (*3497*)	0,0103 (*3605*)
paradigmatic	0,0082 (*7540*)	0,0132 (*7796*)
hierarch.paradigmatic	0,0030 (*1883*)	0,0062 (*1963*)
deriv	0,0007 (*182*)	0,0011 (*187*)
other	0,0023 (*814*)	0,0038 (*841*)
total	0,0123 (*13216*)	0,0242 (*13651*)

Tabelle A.8: Semant. Relationen im Abstand -8 und 8

Relation	Abstand=-9	Abstand=9
A A Gegenteil	0,0006 (*127*)	0,0009 (*127*)
A A Kohyponym	0,0006 (*386*)	0,0008 (*389*)
A A Synonym	0,0001 (*45*)	0,0003 (*46*)
A N typische Eigenschaft	0,0006 (*254*)	0,0007 (*218*)
A V typische Eigenschaft	0,0001 (*40*)	0,0001 (*19*)
N N Beruf zu	0,0003 (*76*)	0,0005 (*112*)
N N Chef von	0,0002 (*55*)	0,0008 (*112*)
N N Eigenname zu	0,0003 (*111*)	0,0009 (*184*)
N N Gegenteil	0,0013 (*235*)	0,0020 (*235*)
N N Grund von	0,0002 (*55*)	0,0005 (*88*)
N N hat Aggregat	0,0002 (*51*)	0,0004 (*60*)
N N hat Beruf	0,0004 (*110*)	0,0005 (*77*)
N N hat Chef	0,0004 (*107*)	0,0003 (*54*)
N N hat Teil/hat Material	0,0008 (*312*)	0,0014 (*356*)
N N hat typischen Ort	0,0018 (*467*)	0,0033 (*607*)
N N Kohyponym	0,0047 (*5066*)	0,0082 (*5239*)
N N Oberbegriff	0,0007 (*483*)	0,0009 (*319*)
N N Synonym	0,0012 (*333*)	0,0017 (*347*)
N N Teil von/Material von	0,0015 (*351*)	0,0022 (*333*)
N N typischer Ort für	0,0012 (*576*)	0,0017 (*478*)
N N Unterbegriff	0,0013 (*312*)	0,0026 (*498*)
N V typisches Objekt/Instrument von	0,0007 (*363*)	0,0027 (*849*)
N V typisches Subjekt von	0,0002 (*99*)	0,0015 (*365*)
V A hat typische Eigenschaft	0,0000 (*19*)	0,0003 (*40*)
V N hat typisches Objekt/Instrument	0,0009 (*841*)	0,0007 (*367*)
V N typische Tätigkeit	0,0008 (*364*)	0,0003 (*106*)
V N typische Tätigkeit für Ort	0,0001 (*31*)	0,0002 (*33*)
V V Gegenteil	0,0002 (*55*)	0,0004 (*55*)
V V Kohyponym	0,0001 (*106*)	0,0003 (*113*)
V V Synonym	0,0001 (*35*)	0,0001 (*35*)
syntagmatic	0,0049 (*3241*)	0,0088 (*3311*)
paradigmatic	0,0069 (*6361*)	0,0114 (*6555*)
hierarch.paradigmatic	0,0030 (*1673*)	0,0053 (*1733*)
deriv	0,0007 (*199*)	0,0011 (*200*)
other	0,0023 (*872*)	0,0041 (*893*)
total	0,0108 (*11652*)	0,0209 (*11968*)

Tabelle A.9: Semant. Relationen im Abstand -9 und 9

Relation	Abstand=-10	Abstand=10
A A Gegenteil	0,0006 (*112*)	0,0007 (*112*)
A A Kohyponym	0,0004 (*339*)	0,0007 (*340*)
A A Synonym	0,0002 (*36*)	0,0002 (*36*)
A N typische Eigenschaft	0,0005 (*248*)	0,0008 (*216*)
A V typische Eigenschaft	0,0001 (*34*)	0,0001 (*24*)
N N Beruf zu	0,0003 (*89*)	0,0007 (*131*)
N N Chef von	0,0001 (*48*)	0,0007 (*105*)
N N Eigenname zu	0,0004 (*105*)	0,0008 (*172*)
N N Gegenteil	0,0011 (*215*)	0,0017 (*216*)
N N Grund von	0,0002 (*52*)	0,0005 (*81*)
N N hat Aggregat	0,0002 (*58*)	0,0003 (*49*)
N N hat Beruf	0,0005 (*127*)	0,0006 (*90*)
N N hat Chef	0,0003 (*102*)	0,0002 (*46*)
N N hat Teil/hat Material	0,0006 (*307*)	0,0016 (*392*)
N N hat typischen Ort	0,0014 (*441*)	0,0031 (*594*)
N N Kohyponym	0,0050 (*5188*)	0,0084 (*5334*)
N N Oberbegriff	0,0008 (*513*)	0,0009 (*318*)
N N Synonym	0,0015 (*375*)	0,0019 (*384*)
N N Teil von/Material von	0,0015 (*395*)	0,0020 (*325*)
N N typischer Ort für	0,0012 (*580*)	0,0017 (*450*)
N N Unterbegriff	0,0011 (*311*)	0,0030 (*522*)
N V typisches Objekt/Instrument von	0,0005 (*292*)	0,0026 (*732*)
N V typisches Subjekt von	0,0002 (*87*)	0,0012 (*310*)
V A hat typische Eigenschaft	0,0000 (*24*)	0,0003 (*34*)
V N hat typisches Objekt/Instrument	0,0008 (*728*)	0,0006 (*300*)
V N typische Tätigkeit	0,0006 (*300*)	0,0004 (*91*)
V N typische Tätigkeit für Ort	0,0001 (*38*)	0,0002 (*38*)
V V Gegenteil	0,0001 (*36*)	0,0002 (*36*)
V V Kohyponym	0,0003 (*106*)	0,0003 (*113*)
V V Synonym	0,0001 (*29*)	0,0002 (*29*)
syntagmatic	0,0044 (*2975*)	0,0085 (*3031*)
paradigmatic	0,0069 (*6423*)	0,0114 (*6581*)
hierarch.paradigmatic	0,0028 (*1762*)	0,0055 (*1801*)
deriv	0,0006 (*179*)	0,0010 (*182*)
other	0,0022 (*856*)	0,0039 (*872*)
total	0,0106 (*11493*)	0,0211 (*11741*)

Tabelle A.10: Semant. Relationen im Abstand -10 und 10

Relation	Abstand=-11	Abstand=11
A A Gegenteil	0,0003 (*93*)	0,0006 (*93*)
A A Kohyponym	0,0004 (*250*)	0,0005 (*251*)
A A Synonym	0,0001 (*35*)	0,0002 (*35*)
A N typische Eigenschaft	0,0005 (*223*)	0,0007 (*191*)
A V typische Eigenschaft	0,0001 (*33*)	0,0001 (*19*)
N N Beruf zu	0,0002 (*78*)	0,0004 (*124*)
N N Chef von	0,0001 (*51*)	0,0007 (*107*)
N N Eigenname zu	0,0004 (*104*)	0,0008 (*148*)
N N Gegenteil	0,0011 (*200*)	0,0016 (*201*)
N N Grund von	0,0002 (*55*)	0,0003 (*71*)
N N hat Aggregat	0,0002 (*47*)	0,0004 (*56*)
N N hat Beruf	0,0005 (*124*)	0,0006 (*81*)
N N hat Chef	0,0003 (*108*)	0,0003 (*52*)
N N hat Teil/hat Material	0,0006 (*313*)	0,0015 (*373*)
N N hat typischen Ort	0,0017 (*429*)	0,0027 (*501*)
N N Kohyponym	0,0041 (*4421*)	0,0072 (*4531*)
N N Oberbegriff	0,0006 (*453*)	0,0008 (*291*)
N N Synonym	0,0014 (*344*)	0,0017 (*348*)
N N Teil von/Material von	0,0015 (*371*)	0,0022 (*326*)
N N typischer Ort für	0,0009 (*488*)	0,0018 (*432*)
N N Unterbegriff	0,0011 (*288*)	0,0027 (*465*)
N V typisches Objekt/Instrument von	0,0005 (*262*)	0,0022 (*663*)
N V typisches Subjekt von	0,0002 (*80*)	0,0009 (*288*)
V A hat typische Eigenschaft	0,0000 (*19*)	0,0003 (*33*)
V N hat typisches Objekt/Instrument	0,0007 (*656*)	0,0005 (*264*)
V N typische Tätigkeit	0,0006 (*284*)	0,0003 (*84*)
V N typische Tätigkeit für Ort	0,0001 (*35*)	0,0002 (*33*)
V V Gegenteil	0,0002 (*43*)	0,0003 (*43*)
V V Kohyponym	0,0002 (*74*)	0,0003 (*77*)
V V Synonym	0,0001 (*26*)	0,0002 (*27*)
syntagmatic	0,0041 (*2675*)	0,0073 (*2711*)
paradigmatic	0,0061 (*5457*)	0,0096 (*5575*)
hierarch.paradigmatic	0,0027 (*1649*)	0,0051 (*1687*)
deriv	0,0007 (*200*)	0,0012 (*201*)
other	0,0022 (*813*)	0,0036 (*825*)
total	0,0095 (*10127*)	0,0183 (*10317*)

Tabelle A.11: Semant. Relationen im Abstand -11 und 11

Relation	Abstand=-12	Abstand=12
A A Gegenteil	0,0004 (*78*)	0,0006 (*78*)
A A Kohyponym	0,0003 (*236*)	0,0005 (*237*)
A A Synonym	0,0001 (*23*)	0,0002 (*23*)
A N typische Eigenschaft	0,0005 (*214*)	0,0006 (*175*)
A V typische Eigenschaft	0,0001 (*22*)	0,0001 (*13*)
N N Beruf zu	0,0002 (*69*)	0,0005 (*104*)
N N Chef von	0,0001 (*52*)	0,0005 (*91*)
N N Eigenname zu	0,0003 (*87*)	0,0008 (*150*)
N N Gegenteil	0,0011 (*181*)	0,0013 (*182*)
N N Grund von	0,0002 (*42*)	0,0004 (*67*)
N N hat Aggregat	0,0002 (*46*)	0,0003 (*45*)
N N hat Beruf	0,0003 (*105*)	0,0004 (*71*)
N N hat Chef	0,0003 (*91*)	0,0003 (*52*)
N N hat Teil/hat Material	0,0007 (*280*)	0,0016 (*336*)
N N hat typischen Ort	0,0015 (*385*)	0,0026 (*483*)
N N Kohyponym	0,0038 (*4403*)	0,0073 (*4490*)
N N Oberbegriff	0,0007 (*419*)	0,0008 (*300*)
N N Synonym	0,0013 (*326*)	0,0016 (*333*)
N N Teil von/Material von	0,0015 (*338*)	0,0018 (*291*)
N N typischer Ort für	0,0009 (*484*)	0,0015 (*392*)
N N Unterbegriff	0,0012 (*293*)	0,0025 (*424*)
N V typisches Objekt/Instrument von	0,0004 (*209*)	0,0020 (*618*)
N V typisches Subjekt von	0,0003 (*84*)	0,0010 (*269*)
V A hat typische Eigenschaft	0,0000 (*13*)	0,0002 (*22*)
V N hat typisches Objekt/Instrument	0,0006 (*611*)	0,0004 (*209*)
V N typische Tätigkeit	0,0005 (*263*)	0,0003 (*90*)
V N typische Tätigkeit für Ort	0,0001 (*29*)	0,0001 (*21*)
V V Gegenteil	0,0001 (*38*)	0,0002 (*38*)
V V Kohyponym	0,0002 (*75*)	0,0002 (*74*)
V V Synonym	0,0001 (*28*)	0,0001 (*29*)
syntagmatic	0,0038 (*2454*)	0,0067 (*2482*)
paradigmatic	0,0054 (*5377*)	0,0097 (*5470*)
hierarch.paradigmatic	0,0027 (*1521*)	0,0049 (*1547*)
deriv	0,0007 (*185*)	0,0010 (*186*)
other	0,0019 (*735*)	0,0032 (*744*)
total	0,0087 (*9649*)	0,0176 (*9790*)

Tabelle A.12: Semant. Relationen im Abstand -12 und 12

Relation	Abstand=-13	Abstand=13
A A Gegenteil	0,0003 (*71*)	0,0005 (*71*)
A A Kohyponym	0,0003 (*186*)	0,0003 (*184*)
A A Synonym	0,0001 (*17*)	0,0001 (*17*)
A N typische Eigenschaft	0,0005 (*194*)	0,0005 (*150*)
A V typische Eigenschaft	0,0001 (*18*)	0,0001 (*13*)
N N Beruf zu	0,0002 (*68*)	0,0003 (*83*)
N N Chef von	0,0002 (*40*)	0,0004 (*76*)
N N Eigenname zu	0,0004 (*82*)	0,0008 (*135*)
N N Gegenteil	0,0009 (*171*)	0,0013 (*171*)
N N Grund von	0,0001 (*46*)	0,0004 (*70*)
N N hat Aggregat	0,0002 (*47*)	0,0004 (*48*)
N N hat Beruf	0,0004 (*84*)	0,0004 (*69*)
N N hat Chef	0,0002 (*75*)	0,0002 (*40*)
N N hat Teil/hat Material	0,0005 (*262*)	0,0014 (*323*)
N N hat typischen Ort	0,0015 (*393*)	0,0025 (*428*)
N N Kohyponym	0,0034 (*3639*)	0,0056 (*3715*)
N N Oberbegriff	0,0005 (*385*)	0,0008 (*293*)
N N Synonym	0,0013 (*293*)	0,0014 (*296*)
N N Teil von/Material von	0,0013 (*325*)	0,0019 (*268*)
N N typischer Ort für	0,0008 (*427*)	0,0015 (*398*)
N N Unterbegriff	0,0014 (*288*)	0,0022 (*391*)
N V typisches Objekt/Instrument von	0,0004 (*184*)	0,0016 (*550*)
N V typisches Subjekt von	0,0002 (*86*)	0,0009 (*244*)
V A hat typische Eigenschaft	0,0001 (*13*)	0,0002 (*18*)
V N hat typisches Objekt/Instrument	0,0006 (*547*)	0,0004 (*185*)
V N typische Tätigkeit	0,0005 (*243*)	0,0003 (*89*)
V N typische Tätigkeit für Ort	0,0001 (*22*)	0,0001 (*21*)
V V Gegenteil	0,0001 (*23*)	0,0001 (*23*)
V V Kohyponym	0,0002 (*69*)	0,0003 (*74*)
V V Synonym	0,0000 (*24*)	0,0001 (*25*)
syntagmatic	0,0034 (*2258*)	0,0061 (*2276*)
paradigmatic	0,0050 (*4468*)	0,0077 (*4549*)
hierarch.paradigmatic	0,0027 (*1428*)	0,0044 (*1444*)
deriv	0,0008 (*209*)	0,0011 (*210*)
other	0,0020 (*677*)	0,0029 (*682*)
total	0,0081 (*8481*)	0,0151 (*8592*)

Tabelle A.13: Semant. Relationen im Abstand -13 und 13

Relation	Abstand=-14	Abstand=14
A A Gegenteil	0,0003 (*61*)	0,0004 (*61*)
A A Kohyponym	0,0003 (*165*)	0,0003 (*168*)
A A Synonym	0,0001 (*21*)	0,0001 (*21*)
A N typische Eigenschaft	0,0006 (*203*)	0,0004 (*131*)
A V typische Eigenschaft	0,0001 (*23*)	0,0000 (*11*)
N N Beruf zu	0,0002 (*60*)	0,0004 (*81*)
N N Chef von	0,0001 (*37*)	0,0005 (*84*)
N N Eigenname zu	0,0004 (*84*)	0,0006 (*114*)
N N Gegenteil	0,0008 (*149*)	0,0012 (*149*)
N N Grund von	0,0002 (*41*)	0,0003 (*60*)
N N hat Aggregat	0,0002 (*40*)	0,0003 (*38*)
N N hat Beruf	0,0003 (*80*)	0,0004 (*59*)
N N hat Chef	0,0003 (*84*)	0,0002 (*37*)
N N hat Teil/hat Material	0,0006 (*245*)	0,0012 (*271*)
N N hat typischen Ort	0,0016 (*368*)	0,0027 (*448*)
N N Kohyponym	0,0033 (*3448*)	0,0054 (*3515*)
N N Oberbegriff	0,0005 (*344*)	0,0006 (*251*)
N N Synonym	0,0013 (*289*)	0,0014 (*292*)
N N Teil von/Material von	0,0012 (*273*)	0,0019 (*251*)
N N typischer Ort für	0,0009 (*448*)	0,0015 (*373*)
N N Unterbegriff	0,0010 (*241*)	0,0018 (*342*)
N V typisches Objekt/Instrument von	0,0004 (*151*)	0,0016 (*491*)
N V typisches Subjekt von	0,0001 (*62*)	0,0009 (*237*)
V A hat typische Eigenschaft	0,0000 (*11*)	0,0002 (*23*)
V N hat typisches Objekt/Instrument	0,0005 (*485*)	0,0003 (*151*)
V N typische Tätigkeit	0,0005 (*233*)	0,0001 (*64*)
V N typische Tätigkeit für Ort	0,0001 (*26*)	0,0000 (*12*)
V V Gegenteil	0,0001 (*27*)	0,0001 (*27*)
V V Kohyponym	0,0001 (*54*)	0,0001 (*54*)
V V Synonym	0,0001 (*30*)	0,0001 (*30*)
syntagmatic	0,0034 (*2122*)	0,0061 (*2141*)
paradigmatic	0,0048 (*4223*)	0,0073 (*4296*)
hierarch.paradigmatic	0,0025 (*1278*)	0,0039 (*1294*)
deriv	0,0006 (*179*)	0,0012 (*179*)
other	0,0019 (*620*)	0,0027 (*623*)
total	0,0077 (*7893*)	0,0140 (*7996*)

Tabelle A.14: Semant. Relationen im Abstand -14 und 14

Relation	Abstand=-15	Abstand=15
A A Gegenteil	0,0002 (*42*)	0,0003 (*42*)
A A Kohyponym	0,0003 (*145*)	0,0003 (*144*)
A A Synonym	0,0000 (*16*)	0,0001 (*16*)
A N typische Eigenschaft	0,0004 (*163*)	0,0004 (*143*)
A V typische Eigenschaft	0,0001 (*25*)	0,0000 (*9*)
N N Beruf zu	0,0002 (*52*)	0,0004 (*86*)
N N Chef von	0,0001 (*36*)	0,0004 (*74*)
N N Eigenname zu	0,0004 (*81*)	0,0007 (*110*)
N N Gegenteil	0,0007 (*124*)	0,0009 (*124*)
N N Grund von	0,0002 (*37*)	0,0003 (*55*)
N N hat Aggregat	0,0002 (*38*)	0,0002 (*39*)
N N hat Beruf	0,0004 (*86*)	0,0004 (*52*)
N N hat Chef	0,0003 (*74*)	0,0002 (*36*)
N N hat Teil/hat Material	0,0006 (*233*)	0,0013 (*300*)
N N hat typischen Ort	0,0014 (*338*)	0,0020 (*355*)
N N Kohyponym	0,0026 (*3127*)	0,0046 (*3185*)
N N Oberbegriff	0,0005 (*359*)	0,0006 (*261*)
N N Synonym	0,0011 (*281*)	0,0015 (*284*)
N N Teil von/Material von	0,0016 (*302*)	0,0018 (*241*)
N N typischer Ort für	0,0008 (*360*)	0,0011 (*345*)
N N Unterbegriff	0,0011 (*250*)	0,0017 (*355*)
N V typisches Objekt/Instrument von	0,0003 (*152*)	0,0011 (*398*)
N V typisches Subjekt von	0,0002 (*61*)	0,0008 (*218*)
V A hat typische Eigenschaft	0,0000 (*9*)	0,0002 (*25*)
V N hat typisches Objekt/Instrument	0,0005 (*396*)	0,0003 (*154*)
V N typische Tätigkeit	0,0005 (*215*)	0,0002 (*60*)
V N typische Tätigkeit für Ort	0,0001 (*22*)	0,0001 (*18*)
V V Gegenteil	0,0001 (*14*)	0,0001 (*14*)
V V Kohyponym	0,0001 (*44*)	0,0001 (*43*)
V V Synonym	0,0001 (*19*)	0,0001 (*19*)
syntagmatic	0,0031 (*1865*)	0,0047 (*1875*)
paradigmatic	0,0039 (*3792*)	0,0063 (*3851*)
hierarch.paradigmatic	0,0025 (*1302*)	0,0039 (*1315*)
deriv	0,0007 (*180*)	0,0010 (*180*)
other	0,0020 (*596*)	0,0026 (*599*)
total	0,0071 (*7247*)	0,0124 (*7325*)

Tabelle A.15: Semant. Relationen im Abstand -15 und 15

Relation	Abstand=-16	Abstand=16
A A Gegenteil	0,0002 (*40*)	0,0003 (*40*)
A A Kohyponym	0,0002 (*129*)	0,0002 (*129*)
A A Synonym	0,0001 (*21*)	0,0001 (*21*)
A N typische Eigenschaft	0,0004 (*158*)	0,0004 (*121*)
A V typische Eigenschaft	0,0001 (*19*)	0,0001 (*9*)
N N Beruf zu	0,0002 (*52*)	0,0003 (*80*)
N N Chef von	0,0001 (*41*)	0,0003 (*65*)
N N Eigenname zu	0,0004 (*83*)	0,0005 (*108*)
N N Gegenteil	0,0008 (*131*)	0,0010 (*131*)
N N Grund von	0,0002 (*33*)	0,0003 (*49*)
N N hat Aggregat	0,0002 (*38*)	0,0003 (*48*)
N N hat Beruf	0,0003 (*80*)	0,0004 (*52*)
N N hat Chef	0,0003 (*65*)	0,0002 (*41*)
N N hat Teil/hat Material	0,0004 (*191*)	0,0012 (*247*)
N N hat typischen Ort	0,0012 (*288*)	0,0020 (*363*)
N N Kohyponym	0,0026 (*2825*)	0,0043 (*2891*)
N N Oberbegriff	0,0006 (*338*)	0,0007 (*247*)
N N Synonym	0,0011 (*244*)	0,0012 (*244*)
N N Teil von/Material von	0,0013 (*248*)	0,0013 (*194*)
N N typischer Ort für	0,0008 (*360*)	0,0010 (*288*)
N N Unterbegriff	0,0010 (*241*)	0,0020 (*336*)
N V typisches Objekt/Instrument von	0,0002 (*107*)	0,0011 (*382*)
N V typisches Subjekt von	0,0001 (*33*)	0,0007 (*195*)
V A hat typische Eigenschaft	0,0000 (*9*)	0,0001 (*19*)
V N hat typisches Objekt/Instrument	0,0004 (*383*)	0,0002 (*110*)
V N typische Tätigkeit	0,0004 (*195*)	0,0001 (*34*)
V N typische Tätigkeit für Ort	0,0001 (*19*)	0,0001 (*16*)
V V Gegenteil	0,0001 (*17*)	0,0001 (*17*)
V V Kohyponym	0,0001 (*41*)	0,0001 (*38*)
V V Synonym	0,0001 (*24*)	0,0001 (*24*)
syntagmatic	0,0029 (*1691*)	0,0047 (*1698*)
paradigmatic	0,0039 (*3455*)	0,0058 (*3521*)
hierarch.paradigmatic	0,0024 (*1155*)	0,0035 (*1162*)
deriv	0,0007 (*168*)	0,0010 (*168*)
other	0,0019 (*584*)	0,0023 (*585*)
total	0,0068 (*6590*)	0,0115 (*6666*)

Tabelle A.16: Semant. Relationen im Abstand -16 und 16

Relation	Abstand=-17	Abstand=17
A A Gegenteil	0,0002 (*38*)	0,0002 (*38*)
A A Kohyponym	0,0001 (*93*)	0,0002 (*93*)
A A Synonym	0,0000 (*14*)	0,0001 (*14*)
A N typische Eigenschaft	0,0004 (*129*)	0,0004 (*122*)
A V typische Eigenschaft	0,0000 (*8*)	0,0001 (*7*)
N N Beruf zu	0,0002 (*53*)	0,0003 (*69*)
N N Chef von	0,0002 (*40*)	0,0003 (*58*)
N N Eigenname zu	0,0003 (*67*)	0,0004 (*87*)
N N Gegenteil	0,0009 (*118*)	0,0008 (*118*)
N N Grund von	0,0002 (*31*)	0,0003 (*58*)
N N hat Aggregat	0,0001 (*28*)	0,0001 (*21*)
N N hat Beruf	0,0004 (*70*)	0,0004 (*54*)
N N hat Chef	0,0002 (*59*)	0,0002 (*41*)
N N hat Teil/hat Material	0,0005 (*180*)	0,0009 (*238*)
N N hat typischen Ort	0,0012 (*285*)	0,0014 (*313*)
N N Kohyponym	0,0023 (*2457*)	0,0038 (*2493*)
N N Oberbegriff	0,0005 (*302*)	0,0005 (*223*)
N N Synonym	0,0010 (*240*)	0,0011 (*240*)
N N Teil von/Material von	0,0014 (*240*)	0,0012 (*184*)
N N typischer Ort für	0,0007 (*312*)	0,0011 (*288*)
N N Unterbegriff	0,0010 (*213*)	0,0016 (*296*)
N V typisches Objekt/Instrument von	0,0002 (*96*)	0,0010 (*325*)
N V typisches Subjekt von	0,0001 (*42*)	0,0006 (*177*)
V A hat typische Eigenschaft	0,0000 (*7*)	0,0001 (*8*)
V N hat typisches Objekt/Instrument	0,0003 (*318*)	0,0002 (*92*)
V N typische Tätigkeit	0,0005 (*174*)	0,0001 (*42*)
V N typische Tätigkeit für Ort	0,0001 (*17*)	0,0000 (*11*)
V V Gegenteil	0,0001 (*12*)	0,0001 (*12*)
V V Kohyponym	0,0001 (*41*)	0,0002 (*38*)
V V Synonym	0,0000 (*15*)	0,0000 (*15*)
syntagmatic	0,0027 (*1492*)	0,0041 (*1505*)
paradigmatic	0,0035 (*3016*)	0,0051 (*3050*)
hierarch.paradigmatic	0,0022 (*1052*)	0,0030 (*1058*)
deriv	0,0006 (*158*)	0,0010 (*158*)
other	0,0018 (*505*)	0,0019 (*507*)
total	0,0061 (*5816*)	0,0101 (*5865*)

Tabelle A.17: Semant. Relationen im Abstand -17 und 17

Relation	Abstand=-18	Abstand=18
A A Gegenteil	0,0002 (*35*)	0,0002 (*35*)
A A Kohyponym	0,0001 (*77*)	0,0002 (*77*)
A A Synonym	0,0000 (*8*)	0,0001 (*8*)
A N typische Eigenschaft	0,0003 (*112*)	0,0003 (*117*)
A V typische Eigenschaft	0,0000 (*12*)	0,0000 (*6*)
N N Beruf zu	0,0002 (*49*)	0,0003 (*81*)
N N Chef von	0,0002 (*33*)	0,0003 (*53*)
N N Eigenname zu	0,0002 (*52*)	0,0005 (*102*)
N N Gegenteil	0,0004 (*89*)	0,0007 (*89*)
N N Grund von	0,0002 (*30*)	0,0002 (*39*)
N N hat Aggregat	0,0002 (*28*)	0,0002 (*33*)
N N hat Beruf	0,0005 (*81*)	0,0003 (*49*)
N N hat Chef	0,0002 (*51*)	0,0002 (*31*)
N N hat Teil/hat Material	0,0005 (*160*)	0,0010 (*208*)
N N hat typischen Ort	0,0013 (*260*)	0,0014 (*285*)
N N Kohyponym	0,0022 (*2258*)	0,0035 (*2292*)
N N Oberbegriff	0,0005 (*273*)	0,0004 (*160*)
N N Synonym	0,0011 (*216*)	0,0012 (*217*)
N N Teil von/Material von	0,0010 (*212*)	0,0011 (*165*)
N N typischer Ort für	0,0006 (*285*)	0,0012 (*262*)
N N Unterbegriff	0,0008 (*157*)	0,0014 (*270*)
N V typisches Objekt/Instrument von	0,0002 (*85*)	0,0009 (*274*)
N V typisches Subjekt von	0,0001 (*28*)	0,0005 (*140*)
V A hat typische Eigenschaft	0,0000 (*6*)	0,0001 (*12*)
V N hat typisches Objekt/Instrument	0,0004 (*270*)	0,0002 (*82*)
V N typische Tätigkeit	0,0004 (*138*)	0,0001 (*28*)
V N typische Tätigkeit für Ort	0,0001 (*19*)	0,0001 (*9*)
V V Gegenteil	0,0001 (*11*)	0,0001 (*11*)
V V Kohyponym	0,0001 (*28*)	0,0001 (*30*)
V V Synonym	0,0001 (*12*)	0,0001 (*12*)
syntagmatic	0,0025 (*1327*)	0,0037 (*1333*)
paradigmatic	0,0031 (*2726*)	0,0046 (*2762*)
hierarch.paradigmatic	0,0019 (*895*)	0,0026 (*896*)
deriv	0,0008 (*152*)	0,0009 (*152*)
other	0,0017 (*483*)	0,0018 (*483*)
total	0,0058 (*5229*)	0,0090 (*5273*)

Tabelle A.18: Semant. Relationen im Abstand -18 und 18

Relation	Abstand=-19	Abstand=19
A A Gegenteil	0,0001 (*17*)	0,0001 (*17*)
A A Kohyponym	0,0001 (*64*)	0,0001 (*63*)
A A Synonym	0,0000 (*7*)	0,0001 (*7*)
A N typische Eigenschaft	0,0003 (*123*)	0,0004 (*122*)
A V typische Eigenschaft	0,0000 (*9*)	0,0000 (*4*)
N N Beruf zu	0,0001 (*33*)	0,0002 (*55*)
N N Chef von	0,0001 (*28*)	0,0002 (*44*)
N N Eigenname zu	0,0003 (*58*)	0,0004 (*77*)
N N Gegenteil	0,0004 (*75*)	0,0006 (*76*)
N N Grund von	0,0002 (*26*)	0,0002 (*39*)
N N hat Aggregat	0,0001 (*22*)	0,0001 (*21*)
N N hat Beruf	0,0003 (*56*)	0,0002 (*34*)
N N hat Chef	0,0002 (*44*)	0,0001 (*28*)
N N hat Teil/hat Material	0,0005 (*152*)	0,0007 (*193*)
N N hat typischen Ort	0,0012 (*215*)	0,0013 (*244*)
N N Kohyponym	0,0020 (*1883*)	0,0027 (*1908*)
N N Oberbegriff	0,0004 (*219*)	0,0004 (*185*)
N N Synonym	0,0009 (*213*)	0,0010 (*214*)
N N Teil von/Material von	0,0011 (*194*)	0,0012 (*154*)
N N typischer Ort für	0,0005 (*242*)	0,0007 (*215*)
N N Unterbegriff	0,0011 (*183*)	0,0010 (*217*)
N V typisches Objekt/Instrument von	0,0002 (*83*)	0,0010 (*278*)
N V typisches Subjekt von	0,0001 (*34*)	0,0005 (*121*)
V A hat typische Eigenschaft	0,0000 (*4*)	0,0001 (*9*)
V N hat typisches Objekt/Instrument	0,0004 (*276*)	0,0002 (*85*)
V N typische Tätigkeit	0,0003 (*119*)	0,0001 (*33*)
V N typische Tätigkeit für Ort	0,0000 (*18*)	0,0000 (*3*)
V V Gegenteil	0,0001 (*10*)	0,0000 (*10*)
V V Kohyponym	0,0001 (*26*)	0,0001 (*26*)
V V Synonym	0,0001 (*14*)	0,0001 (*14*)
syntagmatic	0,0023 (*1232*)	0,0034 (*1239*)
paradigmatic	0,0028 (*2298*)	0,0037 (*2324*)
hierarch.paradigmatic	0,0021 (*856*)	0,0022 (*857*)
deriv	0,0005 (*124*)	0,0008 (*124*)
other	0,0013 (*394*)	0,0015 (*394*)
total	0,0051 (*4563*)	0,0076 (*4597*)

Tabelle A.19: Semant. Relationen im Abstand -19 und 19

Relation	Abstand=-20	Abstand=20
A A Gegenteil	0,0001 (*21*)	0,0001 (*21*)
A A Kohyponym	0,0001 (*59*)	0,0001 (*59*)
A A Synonym	0,0001 (*12*)	0,0001 (*12*)
A N typische Eigenschaft	0,0003 (*109*)	0,0003 (*87*)
A V typische Eigenschaft	0,0000 (*13*)	0,0000 (*2*)
N N Beruf zu	0,0002 (*38*)	0,0002 (*45*)
N N Chef von	0,0001 (*29*)	0,0003 (*47*)
N N Eigenname zu	0,0003 (*60*)	0,0004 (*67*)
N N Gegenteil	0,0004 (*63*)	0,0005 (*63*)
N N Grund von	0,0001 (*15*)	0,0001 (*26*)
N N hat Aggregat	0,0001 (*30*)	0,0001 (*20*)
N N hat Beruf	0,0002 (*45*)	0,0002 (*38*)
N N hat Chef	0,0002 (*46*)	0,0002 (*28*)
N N hat Teil/hat Material	0,0003 (*133*)	0,0006 (*164*)
N N hat typischen Ort	0,0011 (*198*)	0,0013 (*225*)
N N Kohyponym	0,0018 (*1717*)	0,0025 (*1737*)
N N Oberbegriff	0,0004 (*229*)	0,0003 (*151*)
N N Synonym	0,0007 (*161*)	0,0008 (*162*)
N N Teil von/Material von	0,0010 (*164*)	0,0010 (*135*)
N N typischer Ort für	0,0007 (*226*)	0,0008 (*201*)
N N Unterbegriff	0,0008 (*147*)	0,0011 (*226*)
N V typisches Objekt/Instrument von	0,0001 (*69*)	0,0007 (*199*)
N V typisches Subjekt von	0,0000 (*20*)	0,0003 (*101*)
V A hat typische Eigenschaft	0,0000 (*2*)	0,0001 (*13*)
V N hat typisches Objekt/Instrument	0,0002 (*192*)	0,0001 (*65*)
V N typische Tätigkeit	0,0002 (*101*)	0,0001 (*21*)
V N typische Tätigkeit für Ort	0,0001 (*14*)	0,0000 (*7*)
V V Gegenteil	0,0001 (*11*)	0,0000 (*11*)
V V Kohyponym	0,0001 (*16*)	0,0001 (*15*)
V V Synonym	0,0000 (*8*)	0,0000 (*8*)
syntagmatic	0,0022 (*1041*)	0,0030 (*1048*)
paradigmatic	0,0027 (*2066*)	0,0035 (*2086*)
hierarch.paradigmatic	0,0017 (*775*)	0,0022 (*778*)
deriv	0,0006 (*112*)	0,0008 (*112*)
other	0,0013 (*377*)	0,0017 (*378*)
total	0,0048 (*4106*)	0,0072 (*4135*)

Tabelle A.20: Semant. Relationen im Abstand -20 und 20

Anhang B

Kookkurrenzanalyse auf strukturierten Daten

Dieser Teil ist nicht öffentlich.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Delitzsch, d. 27. Juni 2006 Unterschrift