Joseph Birkner
Mat.Nr. 11393024
**Protokolle zum Colloquium (03.07.2017)**

---

Studentin: Katja Bertholdt

*„Das Zipfsche Gesetz“*

Das Zipfsche Gesetz wurde entdeckt von dem Sprachwissenschaftler George Kingsley Zipf, 1902-1950. Dieser studierte und promovierte erst an der Harvard University, und arbeitete dann temporär als Deutschlehrer in Deutschland. Danach kehrte er als Professor für Linguistik an die Harvard University zurück. Neben der Entdeckung des Zipfschen Gesetzes machte sich Zipf auch einen Namen mit seiner bis dahin ungewöhnlichen Initiative, die Linguistik als Naturwissenschaft zu etablieren.

Das Zipfsche Gesetz basiert auf der tieferen Erkenntnis, dass jegliche Lebewesen dazu neigen, ihre Bedürfnisse mit dem geringsmöglichen Aufwand zu befriedigen. Hierzu zählt auch das Kommunikationsbedürfnis mit natürlicher Sprache. Hier gilt es, die Anzahl der notwendigen Wörter so zu optimieren, dass diese für den alltäglichen Sprachgebrauch minimal ist. Da alltägliche Vorgänge äußerst häufig vorkommen, sind auch die Wörter die zur Beschreibung dieser Vorgönge notwendig sind sehr häufig, aber auch sehr inhaltsarm. Weitaus häufiger noch als die Wörter des Alltags sind s.g. Funktionswörter wie Pronomen und Artikel, die zur Konstruktion jedweder natürlichsprachlicher Ausdrücke an sich notwendig sind.

Das Zipfsche Gesetz nun besagt, dass das Produkt der absoluten Häufigkeit *F(w)* eines Wortes *w* mit dem Rang $R_w$ des Wortes in der Reihenfolge aller Wörter der Sprache *L*, absteigend sortiert nach *F(W)*, für alle Wörter aus *L* etwa konstant ist; d.h. $F(w)*R_w \approx c$. Dies entspricht einer logarithmischen Veretilung der Wörter: Skaliert man die Achsen F(w) $F(w)$ *und* $R_w$ logarithmisch, so ergibt sich eine Gerade.

Diese Konstante hängt natürlich noch von der Größe des Korpus, über dem die Zählung der Wörter durchgeführt wurde, ab. Um die Konstante zu normalisieren, muss statt der absoluten Häufigkeit F(w) des Wortes w statt dessen mit der relativen Häufigkeit $f(w)=F(W)/N$ multipliziert werden, wobei N die insgesamte Anzahl an Wörtern in dem zu untersuchenden Korpus ist.

Im Rahmen des Projektes “Deutscher Wortschatz” an der Universität Leipzig wurden umfangreiche Werkzeuge und Daten zur Untersuchung der Häufigkeit von deutschen Wörtern bereitgestellt. Bei solchen Untersuchungen wird die quantitative Bedeutung von Funktionswörtern deutlich: Die häufigsten Wörter “der”, “die”, “und” und “die” vereinnahmen zusammen über 10% aller Wortvorkommen. Die Zipfsche Konstante c für relative Worthäufigkeiten beläuft sich im Deutschen übrigens auf etwa **0,08**. So können weitere Berechnungen angestellt werden: Möchte man zum Beispiel herausfinden, wie viele Wörter in einem Korpus mit ingesamt 222 Millionen Wörtern mindestens 100 Mal vorkommen, so kann man dies mit $c*N/F(w)=R_w \approx 178.000$ berechnen. Möchte man hingegen wissen, wie viele Wörter genau n mal vorkommen, so geht dies mit der Formel $c*N/n - c*N/(n+1)$.

Da die abweichungen von c bei sehr kleinem oder großem Rang oft groß werden, wurde von dem Mathematiker Mandelbrot im eine verfeinerte Variante des Gesetzes erarbeitet. Zipfsche Verteilungen finden sich auch in vielen natürlichen Situationen, zum Beispiel in der Größenverteilung von Städten oder der Verteilung von sozialem Status.

Studentin: Elena Atanasova

*„Multiple Stringsuche: Das Verfahren von Aho-Corasick“*

Es gibt viele Situationen, wo in einer langen Zeichenkette der Länge N Vorkommen einer kürzeren Zeichenkette der Länge M gefunden werden müssen. Der naive Lösungsansatz ist hierbei, für jede Position in N erneut zu schauen, ob diese und die darauf folgenden Positionen mit den Buchstaben in der kürzeren Zeichenkette übereinstimmen. Dieser Algorithmus hat die Laufzeitkomplexität $O(N*M)$.

Wird nach mehreren Wörtern K gesucht, so muss die Laufzeitkomplexität um den Faktor K erweitert werden: sie beträgt dann $O(K*N*M)$.

Eine erste Abhilfe zur Verringerung der Laufzeit bei der Stringsuche schafft der s.g. Knurt-Morris-Pratt Algorithmus. Dieser speichert bei der Mustersuche Informationen über bereits gewonnenne Ergebnisse beim Matching ab, und speichert diese Informatioenn in einer s.g. Sprungtabelle. So ist es nicht mehr notwendig, die Suche nach einer fehlenden Übereinstimmung immer von vorn zu beginnen, wiederholte Vergleiche werden vermieden. Die Laufzeit dieses Algorithmus beträgt $O(K*(N+M))$.

Um die Suche nach mehreren Strings noch effizienter zu machen, entwickelten Alfred V. Aho und Margaret J. Corasick 1975 den s.g. Aho-Corasick Algorithmus. Dieser bildet den Prozess der Mustersuche als einen endlichen Automaten $A=\{Q,\sum,g,f\}$ über Zuständen $Q$, einem Eingabealphabet $\sum$, einer Trie-Übergangsfunktion $g$ und einer Fehlerfunktion $f$ ab. Dieser Automat arbeitet mit einer Laufzeitkomplexität von $O(N)$. Zum Aufbau des Automaten $A$ sei folgendes gesagt:

- Die Übergangsfunktion $g$ entspricht einem Trie. Dieser besitzt einen Ausgangszustand $S$ und eine Menge an Zuständen $Q$, sowie einer Menge an Endzuständen $F$, die eine Untermenge von $Q$ ist. Jede Kante $E$ beschreibt einen Präfix $P$. Um von einem Zustand $Q_1$ via der Kante $E_1$ in den Zustand $Q_2$ zu gelangen, muss der Anfang des bei Zustand $Q_1$ verbleibenden Suffixes des Wortes $w$ genau dem Präfix $P_1$ entsprechen. Danach wird dieser Präfix vom Anfang von $w$ entfernt. Ein Wort wird als erkannt markiert, wenn der nach der Entfernung des Präfixes $P_1$ verbleibende Suffix von $w$ leer ist, und der Zustandsübergang in einen Zustand aus der Menge der Endzustände $F$ erfolgt ist.
- Die Fehlerfunktion $f$ beschreibt nun zusätzliche Kanten $E$ mit leerem Übergangspräfix, die für jeden Zustand $Q_i$ beschreiben, welche anderen Zustände in $A$ genau die längste Sequenz an Präfixen der in Richtung $Q_i$ verweisenden Kanten direkt bis hin zu $Q_i$ ebenfalls direkt vor sich verlangen. Ist zum Erreichen des Zustandes $Q_n$ zum Beispiel die Sequenz von Präfixen $P_1..P_n$ notwendig, so zeigt die Fehlerfunktion genau dann von $Q_n$ auf $Q_m$, wenn der notwendige Präfixpfad für $Q_m$ genau $P_m..P_n$ entspricht, und $n-m$ größer oder gleich $n-w$ aller anderen Pfade $P_w..P_n$ aller anderen Zustände $Q_w$ ist. So bietet die Fehlerfunktion einen effizienten “Ausweg”, wenn $g$ für den aktuellen Zustand und Präfix keinen Folgezustand bietet.

Dieser sehr effiziente Algorithmus findet nicht nur bei der schnellen Dateiinhaltssuche im Unix-Tool fgrep Anwendung, sondern auch beim Vergleich von Bildern, bei der Virensuche, oder auch in der Bioinformatik zur Durchsuchung von DNA.