

Informatik IV
im Sommersemester 2005
Mittwoch, 1. Juni

IBDS Universität Karlsruhe (TH)
Prof. Dr. Alfred Schmitt
Dipl.-Inform. B. Klimmek
Gebäude 50.34, Am Fasanengarten 5

Semester-Programmieraufgabe Teil 2 und Teil 3

Was bisher geschah

Diese letzten beiden Teile der Programmieraufgabe bauen auf dem ersten Teil auf. Teil 2 und auch Teil 3 kann nur als gelöst anerkannt werden, wenn Teil 1 bereits abgegeben und zufriedenstellend bewertet worden ist. (Wenn Sie Teil 2 oder/und Teil 3 nicht bearbeiten, werden Ihnen ggf. die Teilpunkte von Teil 1 trotzdem angerechnet. Umgekehrt ist dies schwer denkbar.)

Vorführung des zweiten und dritten Teils der Aufgabe beim Tutor ist spätestens am **Freitag, 8. Juli 2005**. Erfolgreiche Abgabe wird abermals mit **20 Punkten** je für Teil 2 und Teil 3 vergütet.

Sie haben also den Prototyp eines Indexierers geschrieben, der mittels Hashing die Wörter *eines* Dokuments in Hashzellen einordnet. Sie haben die Wörter gezählt und die Vorkommen jedes einzelnen Wortes im Text.

Dieses Grundgerüst, das Sie bereits erstellt haben, soll nun noch um einige Funktionen erweitert werden.

Was zu tun ist

Der bereits absolvierte erste Teil der Aufgabe stellt das algorithmische Grundgerüst dar, das nun im zweiten Teil zu einer einigermaßen vollwertigen Suchmaschine ausgebaut werden soll, die auf Ihren lokalen Datenträgern (im wesentlichen: der Festplatte) arbeitet.

Hauptsächlich sind dafür die folgenden Erweiterungen zu implementieren:

- Bisher haben Sie nur eine einzige Datei auf Wortvorkommen durchforstet. Nun soll ein ganzer Verzeichnisbaum nach Dateien, die Text enthalten, durchsucht werden.

Informieren Sie sich also über Systemroutinen (aus Standard-Bibliotheken) ihrer Programmiersprache, die das Durchwandern von Verzeichnisstrukturen und das Auslesen einzelner Dateien erlauben.

- Anstatt die Vorkommen eines jeden Wortes bloß zu zählen, sind die Verweise auf die konkreten Vorkommen im Text abzuspeichern.

Überlegen Sie sich (gemäß der Vorlesung “Informatik IV”, Kapitel 2) eine geeignete Datenstruktur: Erstens muß die Datei, in der das Wort auftrat, gemerkt werden; zweitens die Position des Vorkommens innerhalb dieser Datei.

- Schließlich ist – nach der *Vorverarbeitung* des Indexierens – die Abfrageprozedur als eigentliche Suchmaschine zu implementieren.

Sie operiert auf dem schon erstellten Index. Nach Eingabe eines oder mehrerer Wörter (oder auch Phrasen)¹ werden diese Wörter im Index gesucht.

- Außerdem sind wir neugierig und wollen etwas Statistik treiben. (siehe Anmerkungen)

Anmerkungen

Zum Indexierer (**Teil 2**):

- Datenmüll, der erkennbar *nicht* Text ist, soll identifiziert und sodann effizient überlesen werden können, damit nicht absurde “Wörter” den Index aufblähen, die eigentlich nur Byte-Salat darstellen. Dies gilt insbesondere für lange Multimedia-Dateien, die Bild und Ton enthalten ...

Überspringen Sie ggf. die gesamte Datei?

Oder suchen Sie pro Teilintervall noch nach ASCII-Text?

- Bedenken Sie, daß Wörter wie “der”, “die”, “das”, “ist”, “und”, “mit”, “the”, “nicht”, “in” schnell zu enorm langen Verweislisten/-bäumen in den jeweiligen Hash-Zellen führen werden. Eruieren Sie, ob es sinnvoll ist, ab einem (festzulegenden) Schwellwert an Wortvorkommen die Liste einfach zu kappen oder ganz zu verwerfen.

In diesem Fall sollten sie den Eingriff aber im Index vermerken, damit nicht später bei der Suche – etwa nach einer Phrase wie “das ist der Fall” – kein Ergebnis gefunden wird. Die Suche muß also unter Umständen auch auf die Originaldaten zugreifen können.

Statistische Analyse (**Teil 3**):

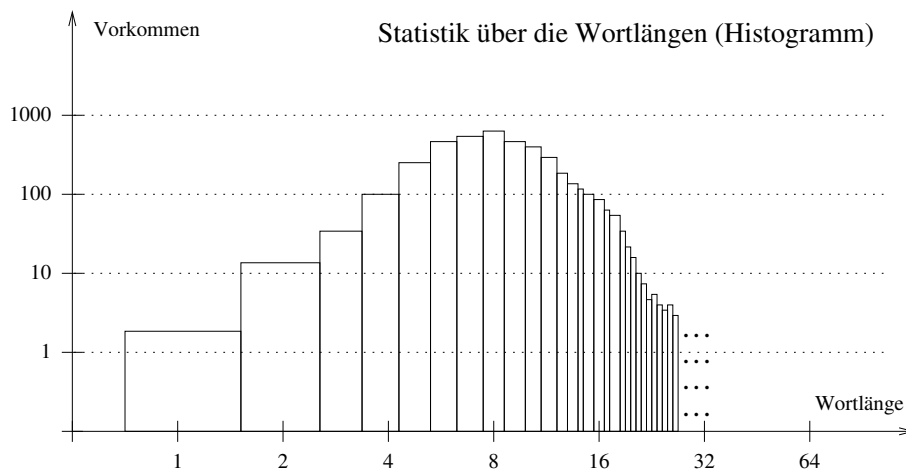
- Nachdem der Indexierer seine Arbeit getan hat, lassen sich vielfältige Statistiken über die im Text (bzw. in den Texten) benutzten Wörter erstellen. In Teil 1 haben Sie ja bereits die Anzahl der verschiedenen Wörter ermittelt, und für jedes Wort die Anzahl seiner Vorkommen.

Diesmal interessieren uns konkret:

- eine interne Statistik über die Qualitäten des Hashings: Wie viele Hash-Zellen sind leer geblieben? Wie viele Hash-Zellen haben genau 1 Eintrag? Wie viele Hash-Zellen haben genau 2 Einträge? usw. Lassen Sie sich grafisch ein Histogramm ausgeben,² das – abhängig vom Wert “Einträge pro Zelle” – die relative Häufigkeit solcher Zellen angibt.
- ein Histogramm über die Anzahl der Wortvorkommen abhängig vom Wort, in absteigender Reihenfolge.
- ein Histogramm über die Anzahl der Wortvorkommen abhängig von der Wortlänge.
- ein Histogramm über die Anzahl der verschiedenen Wörter abhängig von der Wortlänge.

¹Es sollte möglichst sowohl eine UND-Verknüpfung von Suchbegriffen als auch eine Phrasensuche vorgesehen sein. Überlegen Sie sich dazu eine einfache (also nicht überfrachtete!) Eingabesyntax, um diese beiden Fälle bei der Eingabe in die Suchmaschine unterscheiden zu können.

²das können z. B. gängige Tabellenkalkulations-Programme tun, wenn sie mit den entsprechenden Daten gefüttert werden



Damit das Histogramm einigermaßen lesbar und schön wird, ist es voraussichtlich empfehlenswert, die Beschriftung der beiden Achsen *logarithmisch* vorzunehmen, wie hier im Beispiel.

Zur Suchmaschine (**Teil 2**):

- Bei mehreren Suchbegriffen: Die gefundenen Vorkommen müssen sodann daraufhin überprüft werden, ob sie in denselben Dokumenten (also Dateien) aufgetreten sind. Bei Phrasen ist darüberhinaus darauf zu achten, *wo* im Text die Vorkommen stehen.

Aber auch bei UND-Verknüpfung von Suchbegriffen kann es sinnvoll sein, daß die Vorkommen nicht beliebig weit voneinander entfernt in der Datei auftreten.

- Zur Steigerung des Bedienungskomforts Ihrer Suchmaschine ist es unerlässlich, daß Sie die Vielzahl von Suchergebnissen – nach bestimmten Kriterien – ordnen, damit nicht eine “unwichtige” Fundstelle höher gerankt wird als eine essentielle.³ Dies kann beliebig kompliziert veranstaltet werden; wir erwarten hier allerdings keine Geniestreiche, da verzettelt man sich viel zu leicht ...
- Die präsentierten Fundstellen selbst sollten nicht nur den Namen der Datei und die Stelle innerhalb der Datei enthalten; wesentlich ergonomischer ist es, wenn – wie bei herkömmlichen Suchmaschinen – noch der Kontext, in dem die Suchbegriffe stehen, mit ausgegeben wird, bspw. 50 Zeichen vorher und 50 Zeichen nachher.

³Ein paar Ideen finden Sie unter <http://www.suchmaschinentricks.de/ranking/ranking.php3> oder http://www.google.de/why_use.html