Informatik IV

im Sommersemester 2004
Montag, 24. Mai

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

Semester-Programmieraufgabe
Teil 1

Motivation

Die Studierenden im Vordiplom sollen anhand einer nichttrivialen Programmieraufgabe die Anwendung – und die Tücken – von Information-Retrieval-Methoden kennenlernen.

Die Benutzung von Suchmaschinen gehört heute bereits zu den sog. Kulturtechniken wie lesen und schreiben; man kann sich kaum noch an Zeiten erinnern, wo dieser Service zur schnellen und umfassenden Informationsbeschaffung[1] nicht zur Verfügung stand.

Umso interessanter ist zu wissen, was intern geschieht, wenn immense Datenmengen schnell nach bestimmten Inhalten durchsucht werden. Die Vorlesung "Informatik IV" gibt dazu konzeptionelle und auch implementationsstrategische Einblicke in die Techniken der Indexierung.

Organisatorisches

Die Semester-Programmieraufgabe soll in Kleingruppen von höchstens **2 Leuten** in Angriff genommen werden. Sie ist freiwillig zu bearbeiten, nicht Pflicht.

Jede der 2–3 Teilaufgaben, von denen diese die erste ist, ist gesondert abzugeben und dem Tutor vorzuführen. Bei erfolgreicher Bearbeitung gibt es **20 Punkte pro Aufgabenteil**. Sie werden auf die Punkte, die es für abgegebene Übungsblätter gibt, angerechnet.

Vorführung dieses ersten Teils der Aufgabe beim Tutor ist spätestens am **Freitag, 11. Juni 2004**.

Teil der Aufgabe ist, den Studierenden möglichst viele Entwurfs- und Implementationsentscheidungen selbst zu überlassen, sodaß nur die nötigsten Schnittstellen nach außen (Ein- und Ausgabeformate) vorgegeben werden. Dazu gehört insbesondere die Wahl der Programmiersprache; eine Interpretersprache wird aber voraussichtlich nicht schnell genug sein.

Aus Gründen der Einfachheit gehen wir davon aus, daß Sie am liebsten (und besten) zuhause am heimischen PC, am eigenen Laptop oder an Ihrem Stammplatz im ATIS-Pool programmieren. Dort (Laptop oder Pool) sollte auch die Abnahme durch den Tutor stattfinden, mit dem Sie bald einen Termin vereinbaren.

[1] Eigentlich müßte es ja statt "Informationsbeschaffung" Informations-Wiederbeschaffung heißen; wie auch im sehr sehenswerten Film "Brazil" – einer lustigen Anti-Utopie, die tatsächlich im Orwell-Jahr 1984 in die Kinos kam –, wo als Teil einer gigantischen Bürokratie das *Ministerium für Informationswiederbeschaffung* mit einem ebenso gigantischen Personalstab beauftragt ist, keinerlei Informationen verloren gehen zu lassen ...

Was ist zu tun?

Schreiben Sie ein Programm, das zunächst eine einzige Datei indexiert.

Diese Indexierung muß noch nicht vollständig sein; fürs erste genügt es hier in Teil 1, daß alle Wörter, die im Text der Datei vorkommen, in den Index geschrieben werden, und zwar nur genau einmal pro Wort. Das heißt, Ihr Programm muß feststellen, ob ein Wort schon in der Datei vorgekommen ist:

- Wenn nicht, dann muß ein neuer Indexeintrag angelegt werden.
- Wenn ja, so soll ein Zähler, der für jeden Eintrag vorgesehen ist, inkrementiert werden.

Weiterhin soll die Anzahl der insgesamt im Text vorkommenden *verschiedenen* Wörter gezählt werden.

Dabei soll die von Anfang bis Ende des Indexierungslaufs verbrauchte Zeit gemessen werden.

Gegeben also der Name (bzw. Pfad) einer Datei.

Gesucht die Gesamtanzahl der vorkommenden Wörter, die Anzahl der verschiedenen vorkommenden Wörter, ferner die benötigte Zeit zur Index-Erstellung. Außerdem soll für ein nachträglich einzugebendes Wort schnell (d. h. mithilfe des Index) die Anzahl der Wortvorkommen im Text ausgegeben werden können.

Anmerkungen

- Wie auch in der Vorlesung stets betont worden ist, muß auf Effizienz geachtet werden. Dies umfaßt sowohl Speicher- als auch Zeiteffizienz.

  Achten Sie also darauf, daß Ihre Einträge nicht unnötigen Ballast mitführen.

  Überlegen Sie sich ferner, welche Zugriffsmethoden Sie zum Auffinden von Indexeinträgen verwenden wollen. Die Standardmethode wird hier natürlich Hashing sein. Konzipieren Sie am besten selber eine gewiefte Hashfunktion; sie muß in späteren Erweiterungen erheblich mehr Datenmaterial bewältigen als hier in Aufgabenteil 1.

- Ein Problem, das wir hier vernachlässigen wollen, ist die Flexion (Beugung, also grammatische Deklination und Konjugation) von Wörtern. Ihr Programm muß also nicht "`Wort`", "`Worte`", "`Wörter`" und "`Wörtern`" in dieselbe Zelle hashen können, ebensowenig "`kein`" und "`keine`".[2]

- Zu den berüchtigten eigenen Implementationsentscheidungen gehört jene darüber, was als *ein Wort* im Sinne der Indexierung anzusehen ist:

  - Klar dürfte sein, daß durch ein Leerzeichen (Blank, "whitespace") zwei Wörter in der ursprünglichen Datei getrennt werden. Wie gehen Sie mit mehreren Blanks um? Was ist mit Zeilenwechseln, Tabulatoren etc.?
  - Werden Groß- und Kleinschreibung differenziert oder nicht?
  - Wie verfahren mit "`3,14159265`"? Was ist mit Zeichenketten wie "`C++`"? Wie verfahren mit nationalen Sonderzeichen wie "`ß`" oder "`Ü`"?

[2] Wenn dieses Problem mit vertretbarem Aufwand zu bewältigen wäre, wäre dies ja überaus wünschenswert: man gäbe *einen* Vertreter dieser Äquivalenzklasse in die spätere Suchmaschine ein und bekäme alle semantisch gewünschten (aber grammatisch durch lästige Endungen verhunzten) Ergebnisse.

  - Sind "`Index-Erstellung`" ein oder zwei Wörter? Oder macht es Sinn, "`Index`", "`Erstellung`" *und* "`Index-Erstellung`" gleichermaßen zu indexieren?
  - Was ist mit Interpunktionszeichen wie Klammern, Punkten, Kommata etc., die allesamt *nicht* durch Blanks von den Wörtern getrennt sind? Was mit anderen Sonderzeichen?
  - 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 Multimedia-Daten wie Bild und Ton.

  In allen strittigen Fällen sollten Sie eine individuelle Entscheidung treffen, die Ihrer Argumentation von "intelligentem Verhalten" entspricht.

- Erfahrungsgemäß programmieren die einen weniger und die anderen mehr, die einen besser und die anderen schlechter. Insbesondere ist bei (vermeintlicher) Gruppenarbeit dieses Phänomen zu beobachten. Da dies nicht abzustellen ist, möchten wir aber wenigstens, daß auch die Gruppenmitglieder, die nicht schwerpunktmäßig selber programmiert haben, das Programm (den Programmtext!) soweit nachvollziehen und erklären können, daß sie Fragen des Tutors nach Implementationsentscheidungen konkret anhand des Programmtextes beantworten und erläutern können.

- Wir werden auf dem Lernserver demnächst Textdateien moderater Größe anbieten, mit denen Sie Ihre Index-Programme testen können.

- In weiteren Teilen der Programmieraufgabe soll Ihr in Teil 1 begonnenes Gerüst weiter ausgebaut werden.