Information Retrieval

WS 2015 / 2016

Lecture 1, Tuesday October 20th, 2015
(Introduction, Inverted Index, Zipf's Law)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

UNI FREIBURG

Overview of this lecture

- Organizational
  - Contents of this course — demos + list of topics
  - Organization and style — lectures, exercises, tutorials
  - Credits — ECTS points + exam info
  - Coding Standards — valid throughout the course
- Contents
  - Keyword Search — inverted index, Zipf's law

  - **Exercise Sheet 1:** implement keyword search using an inverted index on a collection of 200K movie descriptions

Contents of this Course 1/2

- Three demos for starters

M = million, B = billion

  - **CompleteSearch** Search As You Type

    Data: over 3M publication records from computer science

    Features: suggestions, facets, lightning fast

  - **Broccoli** Semantic Search

    Data: Freebase (2B facts) + Wikipedia (300M sentences)

    Features: search in facts + text, suggestions, fast

  - **Aqqu** Question Answering

    Data: Freebase (2B facts)

    Features: free-form natural language questions

Contents of this Course 2/2

- Research topics behind the demo you just saw
  - Indexing — needed for fast query times
  - Ranking — most relevant hits should come first
  - Compression — lots of data, store it efficiently
  - Error-tolerant search — errors in query or document
  - Web app stuff — JavaScript, AJAX, Cookies, UTF-8
  - Machine learning — solve classification tasks automatically
  - Knowledge bases — how to organize factual knowledge
  - Evaluation — argue that one system is better than another

You will learn about all that (and more) in this course

Organization and Style 1/5

- Organization of the lectures
  - Tuesday 16:15 – 17:45 h in room SR 01-013/019
  - 14 lectures altogether (last one on February 9)

    No lecture on December 29 + January 5 + one other date
  - All lectures are recorded + online by Tuesday evening

    Slides + Audio + Video … Editing: Dennis Weggemann
  - You find all the course materials on our Wiki

    Recordings, slides, code from the lecture, exercise sheets + specifications + design suggestions, master solutions, …

    Also in the SVN, subfolder /public (except for the recordings)

Organization and Style 2/5

- Organization of the exercises
  - One sheet per week, altogether 13 sheets
  - You have one week per sheet

    Until 2 hours before the next lecture = Tuesday 14:00 h
  - You can work in groups of **at most two** people

    If you want to work in a group, send an email to Axel Lehmann (lehmanna@cs.uni-freiburg.de) with the name of your RZ accounts (initials + short number)

    He will then create a joint folder in our SVN for you

    The exercises are the most important part of the course

Organization and Style 3/5

- Organization of the Tutorials
  - There is a **forum** for questions of all kinds

    See the instructions on the back of Exercise Sheet 1

    Response times on the forum are fast, often I or one of the assistants will answer

    Assistants: Elmar Haußmann and Björn Buchhold
  - You will receive **feedback** for each your exercise sheets

    Usually by Friday after the submission deadline

    You will find the feedback in a file feedback-tutor.txt in your subfolder in our SVN

Organization and Style 4/5

- Style of the lectures
  - I will provide: basic definitions, examples, **live code**

    The emphasis is on motivation + the basic ideas

    Working out the details is **your** job in the exercises
  - Underlying theory wherever needed

    No more no less
  - One topic per lecture + self contained

    We provide all the materials you need for the sheets and the exam … the literature pointers at the end are optional

Organization and Style 5/5

- Style of the exercises
  - Your task: understand the basic idea + implement it

    Implementation is great, because it makes you understand all the important details + a working implementation is proof that you did understand it
  - Practically relevant tasks + real data + own experiments

    Usually the best motivation to work on something

    By doing experiments yourself, you will also get a feeling of what research in this area is like
  - Some theoretical tasks

    But not too many

Credits 1/2

- Amount of Work / ECTS points
  - This course yields 6 ECTS points = costs 180 working hours

    Lectures (≈ 30 hours) + exercise sheets + exam preparation
  - Time management options ES = Exercise Sheet

    A. 7-9 hours per ES, little exam prep. **RECOMMENDED**

    B. 5-6 hours per ES, much exam prep. MINIMUM

    C. 0 hours per ES, ??? exam prep. **VERY BAD IDEA**

Doing all the exercise sheets and understanding everything behind them is the perfect preparation for the exam

Credits 2/2

- Exam
  - There is a written exam in the end

    The date will be fixed in one of the last lectures
  - There will be six tasks, out of which you can choose five

    See exams from last years on the Wiki
  - More information about the exam in the last lecture

    We will look at some typical tasks + solve them together

Keyword Search 1/10

- Problem definition
  - Given a collection of text documents ... e.g. the web

    For the exercise sheet: 200K movie descriptions
  - Given a keyword query ... e.g. uni freiburg

    For the exercise sheet: any number of keywords
  - Return all docs that contain at least one of the keywords

    For the exercise sheet: the more keyword matches in a record, the better

    Sounds good, but you will see (when you work on the sheet) that this is not always a good idea

Keyword Search 2/10

- Issues / Refinements
  - Ordering / ranking of the results — Lecture 2
  - Fast query processing — Lecture 3
  - Space consumption — Lecture 4
  - Find variations of the keywords — Lecture 5
  - Search web application — Lecture 6
  - More web stuff + UTF-8 — Lecture 7
  - Synonyms — Lecture 8

  Today (Lecture 1), we start by doing the minimum that is necessary to get a first workable solution

Keyword Search 3/10

- Naive solution
  - Given a keyword query, iterate over all the documents, and identify those that match

    Similar to what the Unix/Linux **grep** command does
  - Actually not so bad for small text collections

    A modern computer can **scan** through 1 GB of text in about half a second

    But already for 100 GB it would be ≈ 1 minute
  - Current web: ≈ 50 billion pages / 2500 TB of text

    Source: www.worldwidewebsize.com ... assuming 50 KB / page

Keyword Search 4/10

- Inverted index
  - For each word, pre-compute and store the **sorted** list of ids of documents / records containing that word

    uni 13, 57, 57, 114, 987, ...

    freiburg 5, 23, 23, 23, 57, 257, ...

  - These lists are called **inverted lists**

    For now, the same id can occur **multiple** times in the same list if the record contains the word multiple times

    Optimization: store pairs like **(57, 2)** or **(23, 3)**

    For the exercise sheet you can do it either way

Keyword Search 5/10

- Query processing, one keyword
  - The inverted list for that keyword already gives us what we want (all docs containing that keyword)

    freiburg 5, 23, 23, 23, 57, 257, ...

Keyword Search 6/10

- Query processing, two keywords
  - Let L_1 and L_2 be the inverted lists of the two keywords
  - We obtain the sorted list of ids for the matches of any of the two keywords by **merging** L_1 and L_2 → R
  - For sorted lists, this can be done in linear time

uni 13, 57, 57, 114, 987, ...

freiburg 5, 23, 23, 23, 57, 257, ...

R 5, 13, 23, 23, 23, 57, 57, 57, 114, ...

Keyword Search 7/10

- Query processing, k > 2 keywords
  - Let L_1 , L_2 , …, L_k be the inverted lists of the keywords
  - We can do a sequence of pairwise merges:

    Merge L_1 and L_2 → L_{12}

    Merge L_{12} and L_3 → L_{123} … and so on
  - Possible optimizations (not needed for the exercise sheet)

    Order the lists such that $|L_1| \leq |L_2| \leq \ldots \leq |L_k|$

    Then the lengths of intermediate results is minimized

    Or: compute a k-way merge in time $O(k \cdot \Sigma_i |L_i|)$

    More about this in a later lecture

Keyword Search 8/10

- Breaking the text into words (tokenization)
  - Conceptually simple: just define a set of characters that belong to words and a set of characters that don't

    Words are then maximal sequences of word characters

    For Exercise Sheet 1, you can simply consider a-z and A-Z as word characters, all others as separators
  - In reality it's a bit more complicated:

    高見 順：娘よりの聞書きにつき誤引用の可能性あり

    Donaudampfschifffahrtskapitängesellschaftsvorsitzender

    Ã–sterreichische GemÃ¼sebrÃ¼he mit KnÃ¶deln^M

    More about UTF-8 and language stuff in Lecture 7

Keyword Search 9/10

- Construction of an inverted index
  - Store in a **map** from strings (words) to arrays of ints (id)
  - Construction algorithm:

    Iterate over all records, numbering them 1, 2, 3, …

    For each record, iterate over all the contained words

    For each word occurrence, add id of current record to respective inverted list (create it, if new word)

    Let's code this together now !

Keyword Search 10/10

- Zipf's Law
  - Let F_n be the frequency of the n-th most frequent word

    Frequency = total number of occurrences in all record
  - Let us plot n on the x-axis and F_n on the y-axis

    Observation: looks like a hyperbola
  - It turns out that $\mathbf{F_n \sim 1 / n^\alpha}$ for some constant α

    Empirical observation, true for most texts and languages

    After George Kingsley Zipf, 1902 – 1950, American linguist
  - Note: $F_n \sim 1 / n^\alpha$ implies $\log F_n \sim - \alpha \cdot \log n$

    We should hence see a (falling) line in the log-log plot

Coding Standards 1/2

- Quick overview
  - Write your code in Python, Java or C++

    I will often (but not always) use Python in the lectures
  - Follow the specifications in the TIP file, if available
  - Follow our coding conventions **at all times**:

    One non-trivial unit test for each non-trivial method

    Adhere to our coding style + document each method

    Use a standard build/make file + make sure everything runs through without errors on our build system

    You find a detailed description on Exercise Sheet 1 … read it **carefully**, this is valid throughout the course

Coding Standards 2/2

- Daphne
  - You find the links to all the relevant information and systems on your **Daphne** page

    Just log in with your regular RZ account and password (your initials + a short number)

References

- Text book

  Introduction to Information Retrieval

  C. Manning, P. Raghavan, H. Schütze

  Available online under http://www.informationretrieval.org

  Good, up-to-date, comprehensive information on the basics

- Wikipedia articles relevant for this lecture

  http://en.wikipedia.org/wiki/Inverted_index

  http://en.wikipedia.org/wiki/Zipf's_law

  Wikipedia articles on basic algorithms stuff are quite good

  However: still no good article on merging lists !