Information Retrieval

WS 2016 / 2017

Lecture 1, Tuesday October 18th, 2016

(Introduction, Inverted Index, Zipf's Law)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of 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	when fixed rule-based approaches fail
– Knowledge bases	how to organize structured data
– 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 HS 026
  - 14 lectures altogether (last one on February 7)

    No lecture on November 1 + December 27 + January 3
  - All lectures are recorded + online by Tuesday evening

    Slides + Audio + Video … Editing: Alexander Monneret
  - 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

    And we make a strong effort to design them properly !

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, usually I or the assistant or one of the tutors will answer

    Assistant for this course: Patrick Brosi
  - 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: motivation, definitions, examples, **live code**

    The emphasis is on the basic ideas + intuition

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

    No theory for the sake of theory in this course
  - 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 1/3

- 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, more exam prep. MINIMUM

    C. 0 hours per ES, xxx exam prep. **IMPOSSIBLE**

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

Credits 2/3

- Exam
  - There is a written exam in the end

    The date will be fixed in one of the last lectures
  - **You need 50 % of the points from the exercises to be admitted to the exam**

    Will be no problem, if you actively follow the course
  - 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

Credits 3/3

- Plagiarism
  - We did not check this in the past but now we do

    It turned out, to our unpleasant surprise, that quite a significant fraction of people copy solutions from others

    This is obviously (a) cheating and (b) pointless additional work for the tutors
  - If we find out that someone copied a solution, even partly or with some modifications after copying, then:

    That person is **not** admitted to the exam

    We enter **Täuschungsversuch** (attempt of deception) into the HISinOne system … you don't want that

Keyword Search 1/10

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

    For the exercise sheet: 189,898 movie descriptions
  - Given a keyword query ... e.g. astronauts moon

    For the exercise sheet: any number of keywords
  - Return all documents that contain all the keywords

    For the exercise sheet: return at most three such documents, the selection is arbitrary

    In Lecture 2, we will also consider returning documents that contain only some of the keywords

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

astronauts	13, 57, 61, 114, 987, ...
moon	5, 23, 57, 63, 114, 257, ...

  - These lists are called **inverted lists**

    For Exercise Sheet 1, each inverted list may contain a particular record id **at most once**, even if the record contains the word multiple times

    Alternative: store pairs of (record id, count) … we will explore this further in Lecture 2

Keyword Search 5/10

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

astronauts	13, 57, 61, 114, 987, ...

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 **both** of the two keywords by **intersecting** L_1 and L_2
  - For sorted lists, this can be done in linear time

astronauts	13, 57, 61, 114, 987, ...	L_1
moon	5, 23, 57, 63, 114, 257, ...	L_2
	57, 114, ...	R

  - The same principle can be used for **merging** the two lists = computing the ids of matches for **any** of the two keywords

    We will explore this further in Lecture 2

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 intersects (or merges):

    Intersect L_1 and L_2 $\rightarrow$ L_{12}

    Intersect L_{12} and L_3 $\rightarrow$ 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 intersect/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:

    初しぐれ猿も小蓑をほしげ也はつしぐれさるもこみのをほしげなり

    Verkehrsinfrastrukturfinanzierungsgesellschaft

    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 (ids)
  - Construction algorithm:

    Iterate over all records, keeping track of the record id

    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 !

    For Exercise Sheet 1, take care that you add each record
    id **at most once** to the same inverted list … and make
    sure that your code still runs **in linear time** !!!

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: $F_n \sim n^{-\alpha}$ implies $\log F_n \sim - \alpha \cdot \log n$

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

Coding Standards 1/3

- 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 Jenkins … see next slide

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

Coding Standards 2/3

- Jenkins
  - Jenkins is our build system, where you can verify that the code uploaded to our SVN indeed **compiles and passes all tests and has zero checkstyle errors**
  - Submissions that do not pass Jenkins without errors will not be graded (= receive zero points)
  - This may sound strict, but is actually quite reasonable:

    Code that does not even compile, fails basic tests, or is badly formatted is a pain to correct for the tutors

    Enforcing minimum standards is standard procedure, e.g. when submitting articles to a conference

    You have enough time + you can ask on the forum

Coding Standards 3/3

- 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 intersecting/merging lists !