universität freiburg

# Databases and Information Systems

## Lecture 1: Introduction, Inverted Index, Zipf's law

**October 14, 2025**

Prof. Dr. Hannah Bast
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
  - Course systems — Daphne, Forum, SVN
- Contents
  - Keyword Search — Inverted index
  - List intersection — The "zipper" algorithm
  - Zipf's law — About word frequencies

Exercise Sheet 1: implement keyword search as explained today for a collection of around 126K movie descriptions

# Contents of this Course 1/2

- Two demos for starters … both made in Freiburg
  - The **QLever** graph database

    Implements the RDF and SPARQL standards (Lecture 6)

    RDF = a data model that is a variant of relational databases

    SPARQL = the query language for RDF (a variant of SQL)

    Let's look at a couple of **example datasets and queries**
  - The **GRASP** question answering system

    An agentic AI, driven by a large language model, that interactively explores a graph database like a human would

    Let's look at an **example question**

# Contents of this Course 2/2

- Research topics behind the demos you just saw
  - Indexing — needed for fast query times
  - Ranking — most relevant hits should come first
  - Databases — data stored in tables
  - Knowledge Graphs — databases + interoperability
  - Error-tolerant search — errors in query or document
  - Web app stuff — HTTP, HTML, JavaScript, UTF-8
  - Embeddings — use matrices and vectors
  - Vector databases and RAG — databases meets AI

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

# Organization and Style 1/6

* To compensate for the overtime

- Lectures
  - Tuesday 14**:05** – 15:**55** h in SR 101-1-9/13

    With one **break** in between at around 15:00 h
  - 13 lectures altogether (last one on February 3)

    No lecture on Nov 4* and Dec 23, Dec 30, Jan 6 (holidays)
  - All lectures are recorded and live-streamed

    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 2/6

- Exercise sheets
  - The exercise sheets are an important part of the course
  - One sheet per lecture (except the last), altogether 12 sheets

    The deadline is always at 12:00 h before the next lecture
  - The exercises are **voluntary**

    That is, you do **not** need 50% of the points to get your Studienleistung … see slide 12

    
  - You can work in **groups** if you like

    But only one of you should commit solutions … details follow
  - **Plagiarism** is still forbidden and **will be punished**

    It's meaningless and costs our tutors work, just don't do it

# Organization and Style 3/6

- Tutorials
  - There will **no** time slots for tutorials

    We have tried it in 100 variants already: only few people come, and those who come are mostly **passive** → it's meaningless
  - There will be **no** weekly Q&A either

    We have tried that, too, same problem
  - If you submit an exercise sheet and you ask for **feedback**, you will receive it by Friday after the submission deadline

    Precondition: you have to write in your **experiences.md** that you want feedback and which kind of feedback you want
  - There is a **forum** for all kinds of questions and you can make **individual appointments** with your tutor → infos in Lecture 2

# Organization and Style 4/6

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

    The emphasis is on: basic ideas, intuition, inspiration

    Working out the details is **your** job … you only really learn stuff by doing it yourself
  - 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
  - We are nice if you are nice

    We invest a lot of work in this course; please make sure that you do your part of the work (learning and paying attention)

# Organization and Style 5/6

- What you should **learn** in this course
  - Two kinds of understanding are important:

    **A**. Understand the concepts **in depth**

    **B**. Understand how to do things **in practice**
  - At the university you often learn deep theory (**A**), but you don't learn where and how to apply it (**B**)
  - At other institutions you often learn how to do things (**B**), but you don't really understand why it works and you are not really able to adapt or extend it to new situations (**A**)
  - In this course, we give **A** and **B** equal importance, you will learn about both, and the exam will also test both

# Organization and Style 6/6

- Master solutions
  - After the deadline for each sheet, the master solutions for this sheet will be published

    On the Wiki and in the SVN under /solutions

    Important: these master solutions are strictly **for your personal use only,** now and in the future

    Under no circumstances may you pass them on to others or let others access them from your machine

# 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
  - Work per exercise sheet, options

    A. **= 0 hours** per ES — if you don't do them

    B. **~ 8 hours** per ES — if you do them, on average

    C. **a lot more** per ES — if you lack basic prerequisites*

    * We always have a fair share of students who have significant gaps in their basic mathematical understanding or in programming

    → if that applies to you, you are still welcome to do this course, and you could learn a lot, but **please be prepared to have to invest much more work and please do not blame us**

# Credits 2/3

- "Studienleistung"
  - All you need to do to get the Studienleistung for this course

    Register on Daphne and register for the exam

    This is not (yet) urgent … the deadline is on **11.01.2026**
  - Nevertheless, you get points for the exercise sheets

    Whoever works on a sheet will probably want to know how well they did (and get valuable feedback)

    Disclaimer: it depends on how many of you actually submit something (we have four tutors and they can only do so much

# Credits 3/3

- "Prüfungsleistung" (Exam)
  - There is a **written** exam in the end

    The date will be fixed in the next months
  - There will be four tasks à 25 points → 100 points max
  - The final grade for your course is linear in the #points:

| | | | | | |
|---|---|---|---|---|---|
| 50 – 54: | 4.0 | 55 – 59: | 3.7 | 60 – 64: | 3.3 |
| 65 – 69: | 3.0 | 70 – 74: | 2.7 | 75 – 79: | 2.3 |
| 80 – 84: | 2.0 | 85 – 89: | 1.7 | 90 – 94: | 1.3 |
| 95 – 100: | 1.0 | | | | |

  - The exam will test **all** of the following three:

    Basic knowledge, practical application, deeper understanding

    Superficial understanding will not be enough to pass

# Keyword Search 1/9

- Problem definition
  - Given a collection of text documents

    For ES1: 126,648 movie descriptions
  - Given a keyword query ... e.g., **spielberg dinosaurs**

    For ES1: any number of keywords
  - Return documents that contain **all** the keywords

    For the exercise sheet: return at most three such documents, see the sheet concerning the selection

    In Lecture 2, we will consider **ranking** as well as returning documents that contain only **some** of the keywords

# Keyword Search 2/9

- 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

    grep takes around 1 GB / s to find all matches of a given string … so not a bad approach for small text collections
  - Google search indexes around 1 trillion pages

    Source: USA vs. Google Antitrust Trial Transcripts

    With an average of 10 kB of text per page, that is around **10 PB** (Petabyte) = 10'000 TB (Terabyte) = 10'000'000 GB

    Fun fact: one way to estimate the number of web pages is based on **Zipf's law**, which we will learn about on slide 24

# Keyword Search 3/9

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

    **spielberg** 6, 22, 24, 53, 62, 102, 148, 194, 197, …

    **dinosaurs** 62, 194, 404, 560, 582, 713, 838, 1053, …
  - These lists are called **inverted lists**

    For Exercise Sheet 1, each inverted list should 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 4/9

- Query processing, **one** keyword
  - If our query consists of only a single keyword, the inverted list for that keyword already gives us what we want

    (namely the ids of all text records containing that word)

    **spielberg** 6, 22, 24, 53, 62, 102, 148, 194, 197, ...

# Keyword Search 5/9

- 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

$L_1$ **spielberg** 6, 22, 24, 53, 62, 102, 148, 194, 197, …

$L_2$ **dinosaurs** 62, 194, 404, 560, 582, 713, 838, 1053, …

$L_1 \cap L_2$ 62, 194, …

  - 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 merging of lists further in Lecture 2

# Keyword Search 6/9

- 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 $L_2$ → $L_{12}$

    Intersect $L_{12}$ 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

    **K-way intersect or merge** in time $O(\log k \cdot \sum_i |L_i|)$

    More about that maybe in a later lecture

# Keyword Search 7/9

- 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

    Semestereröffnungspartyorganisationskomiteevorsitzende

    Ã–sterreichische GemÃ¼sebrÃ¼he mit KnÃ¶deln^M

    More about UTF-8 and language stuff in Lecture 7

# Keyword Search 8/9

- 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 (if new word, create empty list first)
  - **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 9/9

- Zipf's Law

$F_n = C \cdot n^{-\alpha}$

$\log F_n = \log(C \cdot n^{-\alpha}) = -\alpha \cdot \log n + \log C$

  - Let $F_n$ be the frequency of the n-th most frequent word

    Frequency = total number of occurrences in all records

  - 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 $\alpha$

    Empirical observation, true for most texts and languages

    After George Kingsley Zipf, 1902 – 1950, American linguist

  - Note: $F_n = C \cdot n^{-\alpha}$ is equiv. to $\log F_n = -\alpha \cdot \log n + \log C$

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

# Course Systems 1/4

- **Daphne**, our course management system
  - There is a link on the course Wiki, **please register**!
  - In Daphne, you have an overview over the following:
    - Your tutor
    - Your points
    - Link to the Forum see slide 25
    - Link to the SVN see slide 26
    - Link to our Build System see slide 27

# Course Systems 2/4

- **Forum** for this course
  - There is a link on the Wiki and on your Daphne page
  - Please ask whenever something is not clear

    **Please carefully read our guidelines**

    https://ad-wiki.informatik.uni-freiburg.de/teaching/Manuals/AskingOnAForum

    If you ignore these guidelines, you risk getting suboptimal answers or no answer at all. We are very eager to help you, but **you have to help us so that we can help you**
  - Robin or a tutor or myself will usually answer very quickly

    If for some reason you feel that you need help beyond the forum, you can contact your tutor for a **personal meeting**, more about that in Lecture 2

# Course Systems 3/4

- **SVN** = Subversion http://subversion.apache.org
  - A central repository of all course data on our servers
  - Typical operations: svn **add**, svn **commit**, svn **update**
  - Complete history of all previous versions of the files
  - In this course, we will use it for the following:

    Your submissions for the exercise sheets

    The feedback from your tutors

    Slides, code from lecture, exercise sheets, master solutions

    **I will briefly show you how it works now + Robin will record a short video with instructions and upload it in the next days**

# Course Systems 4/4

- Our continuous build system
  - You can (and should) use it to check whether the code that you have uploaded to our SVN passes all checks

    In particular, if it compiles

    If it is free from checkstyle errors

    And if all tests run through without errors

    **Let me also briefly show you this**

# References

- Text book

  **Introduction to Information Retrieval**

  C. Manning, P. Raghavan, H. Schütze

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

  Good and comprehensive information on the **basics**

- Wikipedia articles relevant for this lecture

  https://en.wikipedia.org/wiki/Merge_algorithm

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

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

  Wikipedia articles on basic algorithms stuff are quite good