1. Introduction to Information Retrieval

Prof. Dr. Goran Glavaš

Center for AI and Data Science (CAIDAS)
Fakultät für Mathematik und Informatik
Universität Würzburg

After this lecture, you'll...

- Understand the basic concepts in IR
- Know how to represent and preprocess text for IR
- Understand the generic formalization of IR models

- Know what this course is about (and be glad you've enrolled it :))
- Know which topics we will cover (and hopefully be intrigued by some of them)
- Know what's your part of the job to earn credits

Outline

- **Part one**
  - What is information retrieval?
  - Text representations and preprocessing
  - Generic information retrieval model

- **Part two**
  - About the course
  - Topics
  - Organization

Outline

- **Part one**
  - What is information retrieval?
  - Text representations and preprocessing
  - Generic information retrieval model

- **Part two**
  - About the course
  - Topics
  - Organization

„Retrieval” and „search”

- What is your first association to „information retrieval”?
- What is your first association to „search” (or „search engine”)?

Retrieval and search

What is information retrieval?

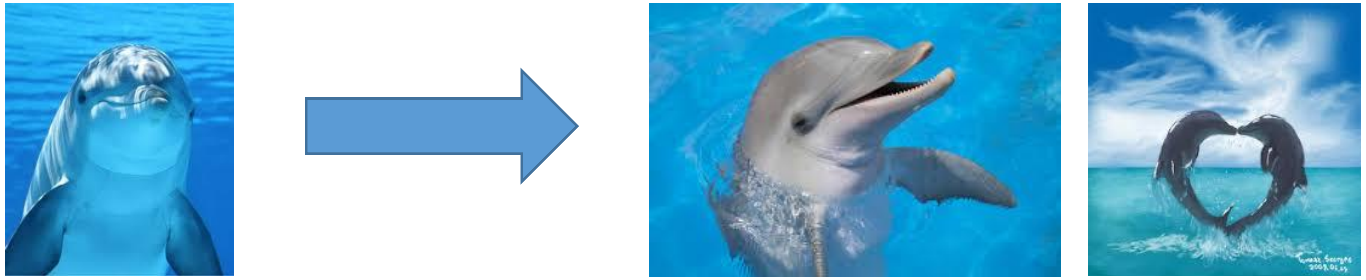
- **Information retrieval** is the activity of obtaining information resources **relevant** for an user’s **information need** from a **collection** of **information resources**.

- Elements of an information retrieval process:
  1. Information needs (users express them in the form of queries)
  2. Information (re)sources, most often unstructured (text, images, video, audio, etc.)
  3. A system/method/model for identifying (re)sources relevant for a given information need (usually from a large collection of information resources)

Information needs

- **Information need** is an individual or group’s desire to locate and obtain information to satisfy a conscious or unconscious need
  - I.e., needs and interests that call for information

- Information needs (conscious or unconscious) are expressed as queries
  - When **retrieving texts**, queries are words or phrases (e.g., „Olympics in London”)
  - In image retrieval queries can also be images



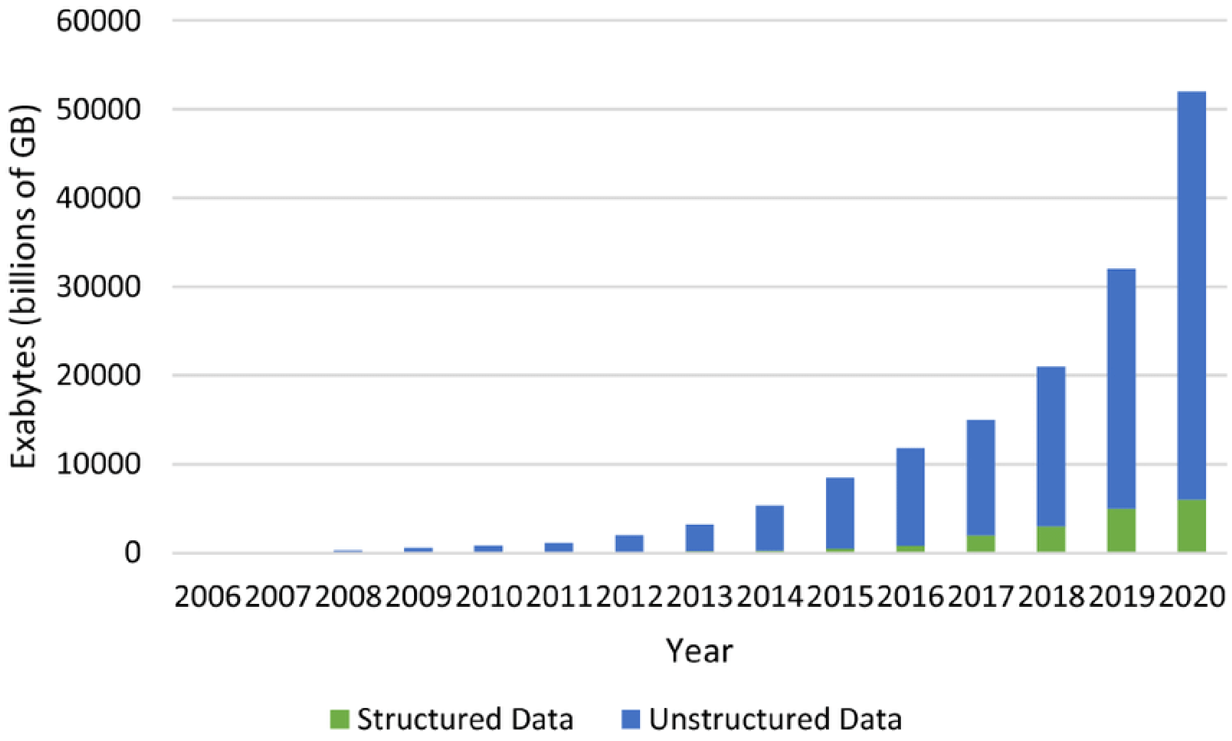
Why (text) information retrieval?

- Because of large repositories of unstructured information sources
  - Companies – tehnical documentation, business documents, contracts, ...
  - Governments – documentation, regulation, laws, ...
  - Science – publications (e.g., Google Scholar)
  - Personal collections – books, emails, files

  - **World Wide Web** – the largest document collection of all
    - Additional challenges due to sheer scale

Why text information retrieval?

- Unstructured sources (text) vs. structured sources (databases)



Why Information Retrieval?

- But do we still need IR in tha age of LLMs?
- Retrieval Augmented Generation (RAG)

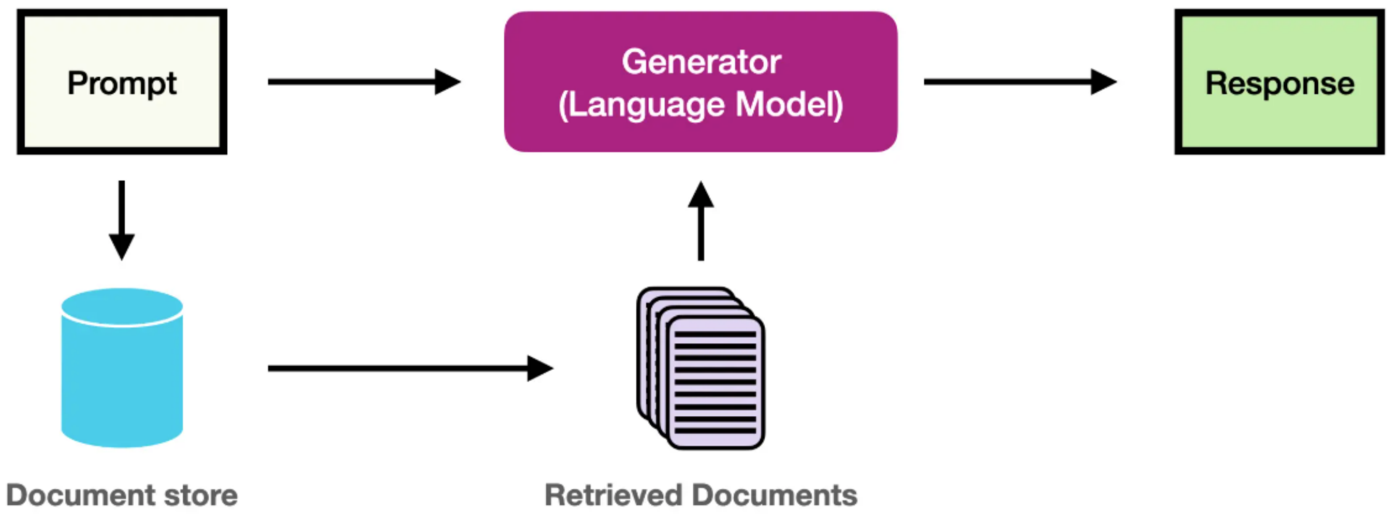


Image from: https://www.promptingguide.ai/research/rag

Text information retrieval

- This course is about **retrieval of text**, where models differ in:
  - Representations of documents and queries
  - Methods for determining (degree of) **relevance** of a document for a given query

- In most IR models relevance is expressed as a score and not a binary decision
  - Documents are ranked in decreasing order according to assigned relevance scores
  - Relevance scores usually incorporate an element of uncertainty

Outline

- **Part one**
  - What is information retrieval?
  - Text representations and preprocessing
  - Generic information retrieval model

- **Part two**
  - About the course
  - Topics
  - Organization

Text representations in IR

1. Unstructured representation

- Text represented as an **unordered set of terms** (the so-called **bag of words**)
- Considerable **oversimplification**
  - We are ignoring the syntax, semantics, and pragmatics of text
  - Is this problematic?

    Q: „Revenue of Apple”
    D: „*Apple* Pencil 2 'to launch in March 2017‘... Microsoft faces drop in *revenue* in the 3rd quarter...”

  - Despite oversimplifying, BoW representations yield good IR performance

- BoW is *de facto* standard IR representation
  - Due to simplicity and speed

Text representations in IR

2. **Weakly-structured representations**
   - Certain groups of terms given more importance – e.g., nouns or named entities
   - Other terms' contribution is either downscaled or completely ignored
   - Some natural language processing (NLP) tools required
     - Part-of-speech (POS) tagger to identify nouns or named entity recognizer (NER) to identify named entities
     - Additional preprocessing can be costly

3. **Structured representations**
   - For example, graphs in which nodes represent some terms/concepts and edges semantic relations between them
   - Sophisticated information extraction (IE) and NLP tools needed to induce structure
   - IE models typically not accurate enough and time-costly
   - Structured representations are virtually not used in IR

Text representations in IR

- Document snippet

„One evening Frodo and Sam were walking together in the cool twilight. Both of them felt restless again. On Frodo suddenly the shadow of parting had falling: the time to leave Lothlorien was near. ”

- Unstructured (bag-of-words) representation

{(One, 1), (evening, 1), (Frodo, 2), (and, 2), (Sam, 1) (were, 1), (walking, 1), (together, 1), (in, 1), (the, 3), (cool, 1), (twilight, 1), (Both, 1), (of, 2), (them, 1), (felt, 1), (restless, 1), (again, 1), (On, 1), (suddenly, 1), (shadow, 1), (parting, 1), (had, 1), (falling, 1), (time, 1), (to, 1), (leave, 1), (Lothlorien, 1), (was, 1), (near, 1)}

Text representations in IR

- Weakly-structured representations
  - Bag of nouns

    {(evening, 1), (Frodo, 2), (Sam, 1), (twilight, 1), (shadow, 1), (parting, 1), (time, 1), (Lothlorien, 1)}

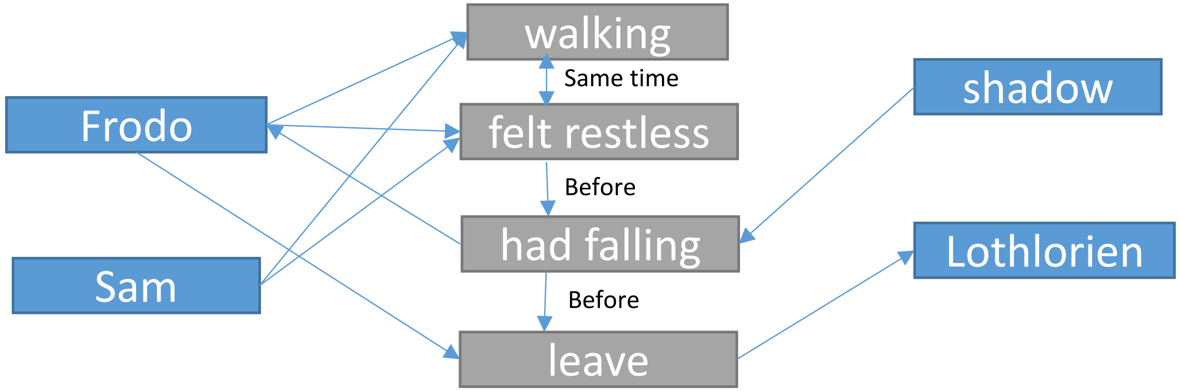
  - Bag of named entities

    {(Frodo, 2), (Sam, 1), (Lothlorien, 1)}

Text representations in IR

„One evening Frodo and Sam were walking together in the cool twilight. Both of them felt restless again. On Frodo suddenly the shadow of parting had falling: the time to leave Lothlorien was near. ”

- Structured representation
  - For example, event-based structure



  - Building such structure requires sophisticated natural language processing tools
  - Structured document representations have not been shown beneficial for IR

Text preprocessing

- So, in IR, we most often use unstructured text representations
  - Text is represented as unordered set of terms (i.e., **bag of words**)

- However, many details about the exact representation are still undefined
  - How do we „split" text into terms? Can this be done in more than one way?
  - Do we consider all terms, or do we want to eliminate some?
    - E.g., functional words that have little meaning like articles and prepositions?
  - How do we treat different forms of the same word?
    - E.g., should „house" be treated the same as „houses"? What about „housing"?
  - What about synonyms or same concepts in different languages?
  - More low-level technical details: what about different document formats?

Text preprocessing

- The **preprocessing** (i.e., preparing text for the retrieval process) usually involves the following steps:
  1. Extracting pure textual content (from HTML, PDF, .docx, OCR-ing images, ...)
  2. Language detection
     - <u>Optional</u> – if you're dealing with multilingual document collections
  3. Tokenization (separating text into character sequences)
  4. Morphological normalization (*lemmatization* or *stemming*)
  5. Stopword removal

- After preprocessing, the text (i.e., the document) is ready to be **indexed**
  - More on indexing in the upcoming lectures

Tokens and terms

- **Word** is a delimited string of characters as it appears in the text
- **Term** is a normalized form of the word (accounting for morphology, spelling, etc.)
  - Word and term are in the same equivalence class – in informal speech they are often used interchangeably
- **Token** is an instance of a word or term ocurring in a document
  - Tokens are „words” in the general sense
  - But numbers, punctuation, and special characters are also tokens
- **Tokenization** is a process, typically automated, of breaking down the text (one long string) into a sequence of tokens (shorter strings)

Tokenization

- Two types of methods for tokenization
  - Rule-based (i.e., heuristic)
  - Based on supervised machine learning models
    - Learn from manually tokenized texts
- Tokenization might seem simple, but it’s not always unambiguous
  - E.g., a simple rule: split string on all whitespaces
    - „Hewlett-Packard declared losses” -> „Hewlett-Packard”, „declared”, „losses”
    - Would we want to split „Hewlett” from „Packard”? What about „lower-case”?
    - What about „Denmark’s mountains”: „Denmark” and „’s”, or „Denmarks”, or „Denmark”?
  - What about tokenizing numbers and punctuation?
    - „19/1/2017”, „55 B.C.”, „+49 176 832 40 332”, „IP: 192.168.0.1”
    - Sometimes spaces are not an indication of an end of a token

Issues in tokenization

- What about different languages?

- German has numerous compounds (Komposita)
  - „Lebensversicherungsgesellschaftsangestellter” (life insurance company employee)
  - Is this a single token or 4 tokens?
  - IR systems for German texts greatly benefit from a compund splitting module

- How about languages that don’t segment text using whitespaces at all?
  - E.g., Chinese
  - „莎拉波娃现在居住在美国东南部的佛罗里达”

Normalization

- Normalization or standardization can involve various changes to the token
  - Error/spelling correction – repairing the incorrect word
  - Case-folding – converting all letters to lower case
    - „Morgen will ich in MIT" – is this German preposition „mit"?
    - Often best to lower case everything (queries and documents)
    - How does Google do it?
      - „C.A.T." (information need: Caterpillar Inc.) returns cat (animal) as the first result
  - Morphological normalization
    - Reducing different forms of the „same" word to a common representative form

Morphological normalization

- **Inflectional normalization** (or **lemmatization**) reduces all lexico-syntactic forms of the same word to one standard form, **lemma** (dictionary headword form)
  - Nouns: singular form in „nominative” case
  - Verbs: infinitive form
  - E.g., „houses” -> „house”, „tried” -> „try”

- **Derivational normalization** reduces all words syntactically derived from some word to the original word (even if the derived word has different meaning)
  - Derivational operators often change the part-of-speech of the word
  - E.g., „destruction” -> „destroy”

- Most IR systems perform inflectional but not derivational normalization

Stemming

- Lemmatization reduces words to dictionary headword entries
  - I.e., the resulting lemma is a string that is again a **valid word** in the language

- **Stemming** is the procedure of reducing the word to its grammatical (morpho-syntactic) root
  - The result of stemming is not necessarily a valid word of the language
    - E.g., „recognized” -> „recogniz”, „incredibly” -> „incredibl”
  - Stemming removes suffixes with heuristics
    - E.g., „automates”, „automatic”, „automation” will all be reduced to „automat”
  - Stemming is „more aggressive” than lemmatization and „less agressive” than derivational normalization
    - „More agressive” means more different words are normalized to the same form

- Stemming is more frequently used in traditional IR systems than lemmatization

Porter’s algorithm

- Most common algorithm for English stemming
- Rule-based algorithm
  - Grammatical conventions and 5 phases of reduction
  - Phases are executed sequentially, one at a time
  - Each phase consists of a set of concurrent suffix-trimming rules
    - If multiple rules apply, use the one that removes the longest suffix

- More on Porter’s stemmer:
  - http://snowball.tartarus.org/algorithms/porter/stemmer.html

- Similar algorithms have been developed for other languages as well

Porter’s algorithm

- Examples of rules
  - „-ing” -> „”
  - „ly” -> „”
  - „sses” -> „ss”
  - „ational” -> „ate”
  - „tional” -> „tion”
- Rules are sensitive to the measure of „how much of a word” a string is
  - Rules consider sequences of consonants and vowels, e.g., $[C][VC]^m[V]$
- Rules also often take into account the length of the remaining „root”
  - E.g., „ement” -> „” is valid only if the remaining word has more than one syllable
    - „replacement” -> „replac” but „cement” -> „cement”

Expansion vs. normalization

- An alternative to normalization is the **expansion** of the query words
  - I.e., we search for alternative forms of the query word as well
- Example
  - **Query:** window **Search:** window, windows
  - **Query:** windows **Search:** Windows, windows, window
  - **Query:** Windows **Search:** Windows
- Theoretically **more powerful** (no need for imperfect normalization)

- In practice **less efficient** as we need to index all words we will be looking for
  - Some languages are highly inflectional and one word can have many different forms
  - E.g., Finnish can have up to 14 different case forms for nouns
    - omena (apple) -> omenan, omenaa, omenaan, omenat, omenien, omenoiden, omenojen, omenain, omenia, omenoita, omenoja, omeniin, omenoihin

Stopword removal

- **Stopwords** are semantically poor terms such as articles, prepositions, conjunctions, pronouns, etc.
- Removal of stopwords is one of the most common steps of IR text preprocessing
- **Q:** Why would we want to remove the stopwords?
  - **A:** Because stopwords have very little meaning, they do not determine whether a document is relevant or not
  - **A:** Removing stopwords reduces the size of vocabulary (and index) and makes retrieval process more efficient
  - **A:** Including stopwords may lead to **false positives** because of stopword matches between query and documents
- Stopword lists for a number of languages:
  - http://www.ranks.nl/stopwords

Outline

- **Part one**
  - What is information retrieval?
  - Text representations and preprocessing
  - General information retrieval model

- **Part two**
  - About the course
  - Topics
  - Organization

General information retrieval model

- We’ve seen what information retrieval is and how to preprocess text
- Now, let’s formalize the general information retrieval model
  - Consider this as a „placeholder” for all concrete IR models we will cover later

- Each functional retrieval system implements the following three components
  1. Representation of a raw query text
     - To be used for matching against documents in the collection
  2. Representation of a raw document text
     - To be used for matching against the query
     - May or may not be the same representation as the one used for query
  3. A function for determining the relevance of documents for the query
     - Taking as input document and query representations – (1) and (2)

General information retrieval model

- Formally, a general retrieval model is a triple of functions (f_d, f_q, r):
  1. f_d is a function that maps documents (raw text) to their representation for retrieval, i.e., $f_d(\mathrm{d}) = \mathrm{p_d}$, where $\mathrm{p_d}$ is the retrieval representation of the document d;
  2. f_q is a function that maps queries (raw text) to their representation for retrieval, i.e., $f_q(\mathrm{q}) = \mathrm{s_q}$, where $\mathrm{s_q}$ is the retrieval representation of the document q;
     - Depending on the IR model, f_d and f_q may or may not be the same function
  3. r is a ranking function which computes a real number indicating the potential relevance of document d for query q, using representations $\mathrm{p_d}$ and $\mathrm{s_q}$:

$$rel(\mathrm{d},\mathrm{q}) = r(f_d(\mathrm{d}), f_q(\mathrm{q})) = r(\mathrm{p_d}, \mathrm{s_q})$$

Index terms and term weights

- **Index terms** are all terms in the document collection (i.e., the vocabulary)
  - Except those we ignore in preprocessing (like stopwords)
  - The set of all index terms: $K = \{k_1, k_2, ..., k_t\}$
  - Each term k_i is, for each document d_j, assigned a weight w_{ij}
  - The weight of the index terms not appearing in the document is 0

- Document d_j is represented by term vector $[w_{1j}, w_{2j}, ..., w_{tj}]$ where t is the number of index terms
- Let g be the function that computes the weights, i.e., $w_{ij} = g(k_i, d_j)$
- Different choices for the weight-computation function g and the ranking function r define different IR models

IR paradigms

- Information retrieval models roughly fall into following paradigms:
  1. Set theoretic models
     - **Boolean model**
     - Extended Boolean model
  2. Algebraic models
     - **Vector space model**
     - Latent models
       - **Latent semantic indexing (LSI)**, Random indexing, Topic modelling for IR
  3. Probabilistic retrieval
     - Classic probabilistic retrieval: **Binary independence model**, BM11, **BM25**
     - **Language models for IR**
  4. Semantic ad-hoc retrieval
     - **Embedding models**
     - **Neural (re)ranking models**

IR paradigms

- Different models are used in the **Web search**
    - Due to sheer size of the Web
    - Because users have no control over the content of the collection
        - **Q:** What is the problem if only content is considered for relevance?
        - **A:** Easy to create spam documents that would be very relevant for certain queries
    - Ranking algorithms that exploit the linked structure of the Web
        - **PageRank**
        - **HITS**

Outline

- **Part one**
  - What is information retrieval?
  - Text representations and preprocessing
  - General information retrieval model

- **Part two**
  - About the course
  - Topics
  - Organization

Course description

- **Q:** Why this course?
  - **A:** Because large collections of unstructured documents from which we retrieve information are all around
  - **A:** Because there are many IR models available, but some are more effective than others in certain settings
  - **A:** Because as information workers and data scientists you are likely to sooner or later have to design/implement a system that retrieves some information from unstructured data collections

- **Purpose of this course**
  - Provide a **systematic overview** of both traditional and advanced methods for text retrieval and web search

Course description

- **Target audience** are students who want to
  - Gain **theoretical understanding** of basic and advanced information retrieval models
  - Obtain **practical (hands-on) experience** implementing IR & WS techniques
- **Prerequisites**
  - Basic knowledge of
    - Linear algebra
    - Probability theory
    - Algorithms and data structures
  - Programming skills in a higher-level programming language
    - E.g., Java, Python, C#, C++
    - Necessary for exercises (and optional bonus projects)
  - Helpful, but not strictly necessary:
    - Knowledge of natural language processing
    - Knowledge of machine learning

Course description

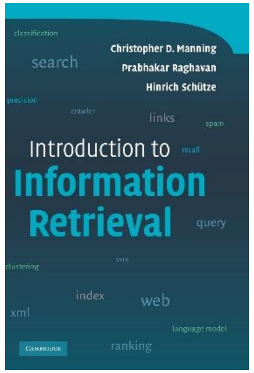
- **What this course covers**
  - Basic theoretical concepts in information retrieval
  - Several traditional information retrieval models
  - Some advanced/recent IR models and techniques
  - IR evaluation
  - Web search and web ranking algorithms

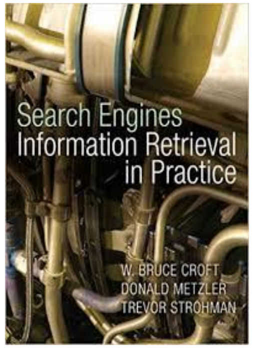
Course description

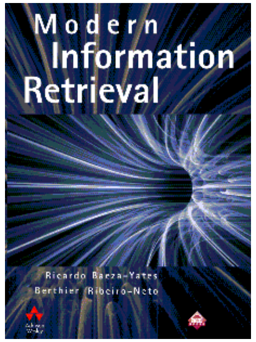
- **What this course doesn’t cover**
  - Natural language processing / Computational linguistics
    - We’ll cover only as much as needed for IR, but won’t go into much depth
  - Machine learning
    - We’ll cover basics needed for IR, but won’t explain the inner workings of ML algorithms
  - Multimedia retrieval (search for images, video, audio)
    - Out of scope, we focus on text retrieval
  - Multilingual and Cross-Lingual IR
    - For those interested in multilinguality → Multilingual NLP

Textbooks

- C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008 (available at http://nlp.stanford.edu/IR-book).
- B. Croft, D. Metzler, T. Strohman, Search Engines: Information Retrieval in Practice, Addison-Wesley, 2009 (available at http://ciir.cs.umass.edu/irbook/).
- R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval, Addison-Wesley, 2011 (2nd Edition).
- Bhaskar Mitra, Nick Craswell: [An Introduction to Neural Information Retrieval](), Now Boston-Delft, 2018.







Course content and schedule

- **Lecture 01**: Introduction to Information Retrieval (April 22)
- **Lecture 02**: Boolean Retrieval and Term Indexing (April 29)
- **Lecture 03**: Data Structures in IR and Tolerant Retrieval (May 6)
- **Lecture 04**: Term Weighting and Vector Space Model (May 13, **Online**)
- **Lecture 05:** Probabilistic IR (May 27)
- **Lecture 06**: LM for IR, Query Likelihood Model (June 3)
- **Lecture 07**: Relevance Feedback and Query Expansion (June 10)
- **Lecture 08**: Latent /Semantic IR Models (June 17)
- **Lecture 09**: Classification & Clustering, Learning to Rank (June 24)
- **Lecture 10**: Neural (Re)Ranking Models, Evaluation (July 1)
- **Lecture 11**: Web Search and Link Analysis (July 8)

Exercises and Project

- There will be **4 exercise sessions**

  - **Exercise #1:** May 14
    - Boolean Retrieval, Indexing, Tolerant Retrieval
  - **Exercise #2:** June 11
    - Vector Space Model, Probabilistic IR
  - **Exercise #3:** June 25
    - Query Exp. & Relevance Feedback, Latent & Semantic Retrieval
  - **Exercise #4:** July 16
    - Learning to Rank, Neural Retrieval, Evaluation & Web Search

  - **Homeworks**, one or two weeks before each exercise session
    - Solutions to be submitted before the session
    - Optional, for the **exam bonus**

(Small) Projects

- **Small-scale projects**, to be carried out in teams of 2-3 students
  - Example topics:
    - Implement (from scratch) the basic vector-space model (VSM) and evaluate its performance on some standard test collection
    - Induce a cross-lingual word embedding space and use it for cross-lingual sentence retrieval
    - Re-rank the results of a traditional retrieval model (e.g., BM25) with a neural relevance model
    - …
  - Optional, for the **exam bonus**
- Project presentations: July 22 (lecture slot)

Examination and grading

- **Final exam**
  - Written exam
  - Exam will asses both **theoretical** and **practical** knowledge
  - **Preparation** for the exam:
    - Exercises
  - 50% of points necessary to pass to course

- **Exam bonus**
  - Increases a passing exam grade by one (e.g., from 2.0 to 1.7)
  - Each homework is evaluated binary: 0 or 1 point (max. 4 points from homeworks)
  - Projects: evaluated on a 0-3 points scale
  - The total of minimally 5 points (out of maximal 7 = 4+3) needed for the bonus

Communication

- This course is powered by WüNLP & CAIDAS
- Your IR & WS teachers
    - Prof. Dr. Goran Glavaš (lecturer)
    - Saad Obaid Ul-Islam (TA)
- **Office hours**
    - As per individual agreements (per email)
- All relevant information will be posted in a **timely** fashion to the WueCampus page

Is this course hard?

- To an extent, this depends
  - On your previous knowledge (linear algebra, probability theory, NLP, ML, ...)
  - On your programming skills (for the project)

- But primarily this depends on
  - Your interest in the IR & WS topics
  - Your enthusiasm and willingness to learn new stuff
  - **The amount of time and effort you invest into this course**

- This course is **5 ECTS** credits
  - One credit should amount to **25-30** hours of your time

Can I pass this course?