



ifis

Institut für Informationssysteme
Technische Universität Braunschweig

Information Retrieval and Web Search Engines

Wolf-Tilo Balke
Muhammad Usman

Institut für Informationssysteme
Technische Universität Braunschweig



Recap: Inverted Indexes

- In **Boolean retrieval**, queries have been evaluated using **inverted indexes** (aka inverted files)
- Document collection:
 - Document1 = {step, mankind}
 - Document2 = {step, China}
- Inverted index:
 - step: {Document1, Document2}
 - mankind: {Document1}
 - China: {Document2}
- Query:
 - “mankind AND step”





Indexing:

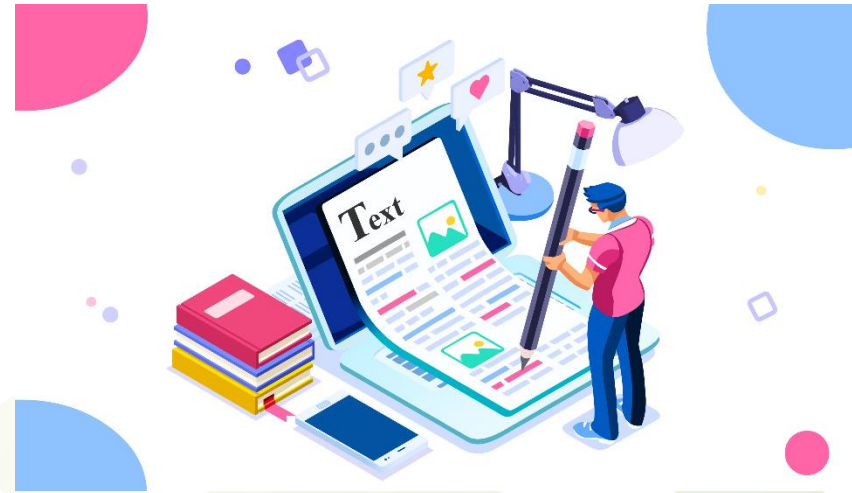
“The process of assigning keywords to each document”

- These **keywords** are used as a (possibly intermediate) **representation** of each document
- **Some problems** we are faced with:
 - “3/12/91” vs. “Mar 12, 1991” vs. “12/3/1991”
 - “<h1>Document heading</h1>” vs. “Document heading”
 - computer vs. computers vs. Computer vs. computer’s
 - aren’t vs. are not



Today's Lecture: Indexing

1. Document Preparation
2. Index Construction
3. Query Evaluation





Document Preparation

INPUT DOCUMENT



DOCUMENT TEXT

Y2K Around the World
As computers all over the world switched to 2000, few Y2K bugs were reported in several labs. [...]



TOKENIZATION

y2k around the world as computers all over the world switched to 2000 few y2k bugs were reported in several labs [...]



FILTRATION

y2k world computers world switched 2000 y2k bugs reported labs [...]



STEMMING

y2k world computer world switch 2000 y2k bug report lab [...]



DOCUMENT REPRESENTATION

y2k (2), world (2), computer (1), switch (1), 2000 (1), bug (1), report (1), lab (1)

Character sequence decoding

Document delinearization



I. Character Sequence Decoding



DOCUMENT TEXT

Y2K Around the World

As computers all over the world switched to 2000, few *Y2K bugs* were reported in several labs.
[...]

- Let's assume some document has to be indexed
- **First step: Getting a textual representation**
- Sounds easy, but might be pretty complicated
 - Many different document formats:
DOC, plain text, PDF, HTML, XLS, PPT, RTF, XML, ...



I. Character Sequence Decoding

- Many document formats can be converted into a **plain text representation** (possibly with some markup information) using special **converters**
- **Example:** pdftohtml (open source tool)



Information Retrieval and Web Search Engines

Lecture 8: Support Vector Machines
January 7, 2009

Wolf-Tilo Balke with Joachim Selke
Institut für Informationssysteme
Technische Universität Braunschweig, Germany

Homework: Exercise 14a

- Given a collection, a query, and an IR system:
 - Collection: **20 relevant** documents, **180 non-relevant**
 - Found: **8 relevant** documents, **10 non-relevant**
- Precision, recall, and fallout?

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{items retrieved})} \quad 8 / 18 \approx 0.44$$

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} \quad 8 / 20 = 0.4$$

$$\text{Fallout} = \frac{\#(\text{nonrelevant items retrieved})}{\#(\text{nonrelevant items})} \quad 10 / 180 \approx 0.06$$

Information Retrieval and Web Search Engines — Wolf-Tilo Balke with Joachim Selke — Technische Universität Braunschweig 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><HTML> <HEAD> <TITLE></TITLE> </HEAD>
<BODY> <A name=1></a><b>Homework: Exercise 14a</b><br> • Given a
collection, a query, and an IR system:
  - Collection: 20 relevant documents, 180 non-relevant
  - Found: 8 relevant documents, 10 non-relevant
  - Web Search Engines
  • Precision, recall, and fallout?
  - Lecture 8: Support Vector Machines
  - January 7, 2009
  - wolf-Tilo Balke with Joachim Selke
  - Institut für Informationssysteme
```



I. Character Sequence Decoding

- **But even plain text can be problematic**
- A plain text document is a **sequence of bytes**
- What does the following byte sequence mean?

102 195 164 104 114 116

- This depends on the **character encoding**
 - A character encoding assigns **byte sequences to characters**

- It means:

- “fährt” in **UTF-8**
- “fÃ¤hrt” in **ISO-8859-1**
- “f❖❖hrt” in **ASCII**
(ASCII is a 7-bit character encoding!)

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x



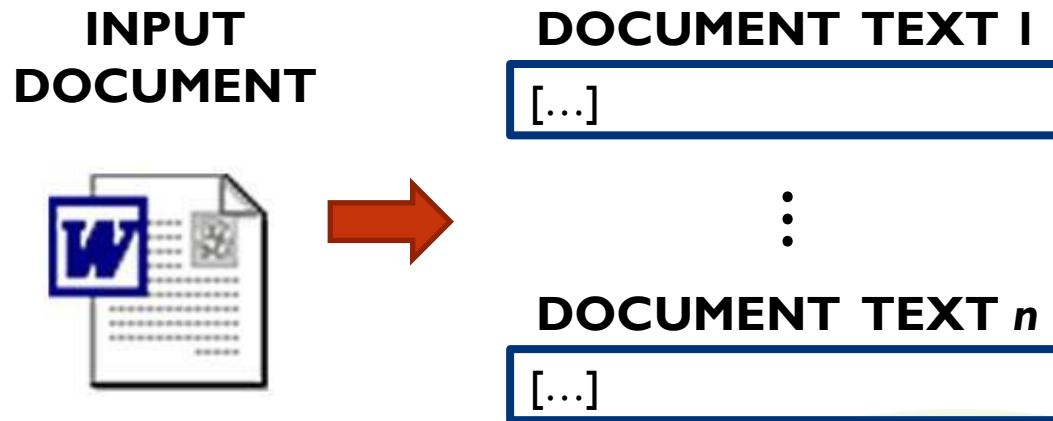
I. Character Sequence Decoding

- Unfortunately, a text document's **character encoding** often is **unknown** or **wrongly specified**
- There are many **heuristics** for auto-detecting encodings:
 - **Coding Scheme Method:**
Exclude certain encodings by looking for **illegal bytes or byte sequences**, which are not defined within this encoding
 - **Character Distribution Method:**
Use **statistics** to exploit the fact that in any given language, some characters are used more often than other characters
 - **Two-Char Sequence Distribution Method:**
Exploit statistical information about **2-grams**, i.e. look at **pairs of adjacent characters**

Source: <http://www.mozilla.org/projects/intl/UniversalCharsetDetection.html>



2. Document Delinearization



- Sometimes, the documents to be indexed are **very large** and should be **split up into smaller parts**
- **Examples:**
 - E-books (large PDFs)
 - E-mail collections, including attachments (e.g. in UNIX's mbox format)
- Again, this normally can be done using **heuristics...**



3. Tokenization

DOCUMENT TEXT

Y2K Around the World

As computers all over the world switched to 2000, few *Y2K bugs* were reported in several labs.
[...]



TOKENIZATION

y2k around the world as computers all over the world switched to 2000 few y2k bugs were reported in several labs [...]

- **Tokenization:**

- Remove formatting information (e.g. HTML tags)
- Remove punctuation
- Carry out basic normalization (e.g. remove capitalization)
- **Goal:** Convert the text into a **sequence of “tokens”**



3. Tokenization

- **Tokenization is difficult!**
- Let's tokenize the following sentence!
“Mr. O’Neill thinks that the boys’ stories about Chile’s capital aren’t amusing.”
- One token or two?
 - Hewlett-Packard
 - State-of-the-art
 - Data base
 - San Francisco
 - York University vs. New York University



3. Tokenization

- Identical tokens (Telephones)?
 - (0531) 391 3271
 - +49 531 391-3271
 - 531.391.3271
- Identical tokens (Dates)?
 - 4/15/99
 - 15/4/99
 - Apr 15, 1999





3. Tokenization

- Specific problems in other languages:
 - This two **Chinese characters** can be treated as one word meaning “**monk**” or as sequence of two words meaning “**and**” and “**still**”

和尚

- How to handle compounds?
 - Donaudampfschiffahrtsgesellschaftskapitänsfrau
 - Lebensversicherungsgesellschaftsfachangestellter



3. Tokenization

- **Normalization** handles most of the problematic cases
- Define **equivalence classes** of character sequences that get mapped to the same token
 - U.S.A. and USA
 - naïve and naive
- Define these classes implicitly by **transformation rules**
 - **Omit all accents**
 - **Remove periods** between two characters, where there is no whitespace around (e.g. in U.S.A.)
 - **Do case folding**, i.e. reduce all letters to lower case
 - Maybe you need exceptions for names: windows \neq Windows (Not important, since users ask **queries in lowercase** anyway)...



4. Filtration

TOKENIZATION

y2k around the world as
computers all over the
world switched to 2000 few
y2k bugs were reported in
several labs [...]



FILTRATION

y2k world computers world
switched 2000 y2k bugs
reported labs [...]


- Removal of **stop words!**
- **Stop words:**
Extremely common words, which are of little value in selecting which documents match a user's query
- **Examples:** a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, which, will, with
- “to be or not to be”?



4. Filtration

- In classical IR systems, stop words have been rigorously deleted
- But stop words are needed for **phrase queries**, e.g. “King of Finland” or “As We May Think”
- For example, **Google** does not remove stop words:

Web [Images](#) [Maps](#) [News](#) [Shopping](#) [Mail](#) [more](#) ▼ [Sign in](#)

 [Advanced Search](#)
[Preferences](#)

Web Results 1 - 10 of about 19,400 for "[king](#) of [finland](#)". (0.18 seconds)

[Monarchy of Finland - Wikipedia, the free encyclopedia](#)
(Redirected from **King of Finland**). Jump to: navigation, search ... of the late king Charles XII of Sweden, could be proclaimed as the **King of Finland**. ...
en.wikipedia.org/wiki/King_of_Finland - 28k - [Cached](#) - [Similar pages](#)

[List of Finnish monarchs - Wikipedia, the free encyclopedia](#)
5 Jan 2009 ... During Svinhufvud's regency, Prince Frederick Charles of Hesse was elected as the **King of Finland** on October 9, 1918. ...
en.wikipedia.org/wiki/List_of_Finnish_rulers - 46k - [Cached](#) - [Similar pages](#)
[More results from en.wikipedia.org »](#)

[Finland's King - TIME](#)
The old man was Jean Julius Christian Sibelius, most famous of present-day composers and "Uncrowned **King of Finland**"; the occasion was his seventieth ...
www.time.com/time/magazine/article/0,9171,758541,00.html - 37k - [Cached](#) - [Similar pages](#)



4. Filtration

- A **general strategy** for stop words removal:
 - Sort tokens by **collection frequency**
 - Take **top-k** of this list as stop words
- Alternatively, use a (possibly domain-specific) **predefined stop word list**
- But:
 - One can handle large indexes by exploiting the **statistics of language** for compression, so the cost for including stop words is not high for modern systems
 - **The trend goes to smaller stop word lists**, e.g. 200–300 or even less



5. Stemming

FILTRATION

y2k world computers world
switched 2000 y2k bugs
reported labs [...]



STEMMING

y2k world computer world
switch 2000 y2k bug report
lab [...]

- **Lemmatization:**

The process of grouping together the different **inflected forms** of a word, e.g. walking → walk, better → good

- **Stemming:**

Trying to do lemmatization using crude heuristics, usually without knowledge of the word's context or any rules of grammar, e.g. walking → walk, but better → better



5. Stemming

- Of course, lemmatization would be the “**right**” thing and there are software tools for this task
- But:
 - “Good” lemmatization is **computationally expensive** if very large document collections are to be processed
 - **Gains** of lemmatizers over stemmers in retrieval quality (mean average precision) **are very modest** for English (0–5%)
 - Larger gains have been reported for other languages, e.g. German, Spanish, and Finnish (10–30%)
- In this lecture, we will discuss only **stemming**



5. Stemming

- The most common stemmer is the **Porter stemmer** (Porter, 1980)
- It is designed to fit the characteristics of **English** language
 - **Idea:** Suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes
- How does it work?
 - The algorithm runs through **five steps**, one by one
 - In each step, several **rules** are applied that change the word's **suffix**





5. Stemming

- Some (simplified!) examples of rules used in the Porter stemmer:

Rule	Example
SSES → SS	caresses → caress
IES → I	ponies → poni
S →	cats → cat
ING →	motoring → motor
Y → I	happy → happi
ATIONAL → ATE	relational → relate
FULNESS → FUL	hopefulness → hopeful
ICAL → IC	electrical → electric
ABLE →	adjustable → adjust
ATE →	activate → activ



5. Stemming

- Some transformations made by the Porter stemmer:

Input word	Stemmed word
gen	gen
gender	gender
genders	gender
general	gener
generally	gener
generals	gener
generation	gener
generations	gener
generative	gener
generosity	generos
generous	gener
genitive	genit
genitivo	genitivo



5. Stemming

- **A comparison of different stemmers:**
 - **Sample text:**

Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation
 - **Porter stemmer:**

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation
 - **Lovins stemmer:**

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation
 - **Paice stemmer:**

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation



Further Normalization

FILTRATION

y2k world computers world
switched 2000 y2k bugs
reported labs [...]



STEMMING

y2k world computer world
switch 2000 y2k bug report
lab [...]

- There are **some additional** things apart from stemming that **could** be done during this step
- Take care of **umlauts** and **accents**
 - Usually, just remove them (e.g. ä → a)
or transliterate them (e.g. ä → ae)
 - But be careful: unbeschränkt ≠ unbeschränkt
- Take care of **synonyms**, e.g. auto → car
 - Again, be **very careful** when doing this!



6. Document Representation

STEMMING

y2k world computer world
switch 2000 y2k bug report
lab [...]



DOCUMENT REPRESENTATION

y2k (2), world (2),
computer (1), switch (1),
2000 (1), bug (1), report (1),
lab (1)

- Finally, we arrive at the **bag-of-words representation**
- The preparation of original documents is finished now, but we need to talk about **efficient data structures for managing the inverted indexes...**



- Preparing documents is particularly problematic
 - A lot of information is encoded in **figures**
 - Text **references** figures and figures may contain **tabular data**
 - **Names** (e.g. of substances) implicitly contain **structural information**
- PDF (portable document format)
 - The **standard for exchanging** digital documents
 - ISO 15930, 19005, 24517, 32000 etc.
 - Documents are **collections of objects**
(characters, vector-based graphics, bitmap images, ...)
 - Objects are positioned by **absolute coordinates**
 - Non-digital documents, scanned or photographed





- The preparation process (for PDFs):
 - Extracting **text**
 - Recognize **entities, relations** within text, tables, and figures
 - Derive **structural data** from named entities





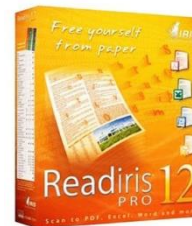
Solution: Extraction Tools?

Detour

- **PDF-to-text converters** are of no help
 - Only extract the text objects from PDF
 - pdftotext, PDFBox, PDFExtractor, PDFTextStream



- Screen reader and **OCR software**
 - Input can be arbitrary → bitmap images
 - Tries to find regions, lines, words, and characters in the image





- “Perfect” documents can be processed easily
 - Single column, well-structured
 - Direct output of the authoring or production process
 - Or converted from other digital formats (XML, Word, LaTeX, ...)
 - Not much markup
- The only problem: **segmentation!**

Introduction

Triterpenes are an important class of plant secondary metabolites derived from C₃₀ precursors [1,2]. More than 100 triterpenoids with different skeletons and functional groups have been isolated and structurally characterized. These may not be essential for the life of the plant, but play an important role in self-defence against harmful organisms and coloring petals and fruits etc., and medicinal uses of these materials are known [3,4]. Recently, triterpenoids have been recognized as *renewables* in supramolecular chemistry and nanoscience [5-10]. Even though the nano-sized triterpenic acids are available in abundance from a variety of plants, a major difficulty in their use is their availability in pure form. Occurrence of the pentacyclic triterpenoids having a β -amyrin skeleton with certain amounts of α -amyrins is common in nature and has been explained by 1,2-CH₃ migration during their biosynthesis [11]. The mixture of the triterpenic acids extractable from *Terminalia arjuna* contains arjunoic acid as the major component along with asiatic acid, as a minor component having a close structural resemblance [12,13]. Biotransformation of the ursane to the oleanane skeleton has recently been reported [14], but no simple method for the separation of the two triterpenic acids is known [15]. Herein we report a simple method for separation the two nano-sized triterpenic acids along with the self-assembly property of arjuna-bromolactone in organic solvents and its 1D-helical structure in the solid state.

Results and Discussion

The mixture of the triterpenic acids **1** and **2** obtained from *Terminalia arjuna* (see Supporting Information File 1) was transformed to a mixture of arjuna-bromolactone **3** and unchanged asiatic acid (**2**) on reaction with bromine in acetic acid, using the



Beilstein Journal of Organic Chemistry 2008, 4, No. 20.

Figure 1: DPP-IV inhibitors.

the N-terminus [14,15]. Thus inhibition of DPP-IV extends the half-life of endogenously secreted GLP-1, which in turn enhances insulin secretion and improves the glucose tolerance. DPP-IV inhibitors offer several potential advantages over existing therapies including decreased risk of hypoglycemia, potential for weight loss, and the potential for regeneration and differentiation of pancreatic β -cells [1].

Because of its key role in DPP-IV inhibition the 2-(S)-cyanopyrrolidine moiety has been found to be an integral part of many DPP-IV inhibitors (Figure 1). Apart from behaving as a proline mimic, the presence of the nitrile on the five-membered ring provides (i) reversible and noncovalent inhibition of DPP-IV and (ii) chemical stability adequate for oral administration (Figure 2).

Figure 2: Role of 2-(S)-cyanopyrrolidine moiety in DPP-IV inhibition.

Chemically, incorporation of a 2-(S)-cyanopyrrolidine moiety into a molecule can be carried out by using (S)-1-(2-chloroacetyl)pyrrolidina-2-carbonitrile (**6**) as a reactant. Thus, compound **6** has become a widely used key intermediate for the synthesis of many DPP-IV inhibitors including NVP-LAF237 (**2**) that are presently under various stages of clinical evaluation [16-23]. For the development of novel DPP-IV inhibitors under our new drug discovery program, we have needed this intermediate (**6**) in bulk quantity. Synthesis of this compound however involves the use of expensive L-prolinamide (**5**) [6,24,25] (Scheme 1), preparation of which in turn requires an N-protect-

Scheme 1: Earlier route to (S)-1-(2-chloroacetyl)pyrrolidine-2-carbonitrile (**6**):

5 (L-prolinamide) $\xrightarrow[1. \text{CICH}_2\text{COCl}]{\text{N-protection/deprotection}}$ 6 (82%)

2. TFAA, RT, 1 h
side workup

Results and Discussion

In our strategy, we decided to use L-proline in place of L-prolinamide because of its easy availability. Additionally, we anticipated that the chloroacetyl group (which is also a part of the target intermediate) might play the role of a protecting group so that the use of an additional protecting group and its removal (i.e., deprotection) can be avoided. Thus L-proline (**7**) was N-acylated with chloroacetyl chloride in refluxing THF to afford 1-(2-chloroacetyl)pyrrolidina-2-carboxylic acid (**8**) (Scheme 2). While preparation of this compound has been reported earlier [26], we encountered several difficulties when following the reported process, the major one being the longer reaction time (48 h) at a low temperature (-20 °C). Nevertheless,

Scheme 2: Synthesis of (S)-1-(2-chloroacetyl)pyrrolidine-2-carbonitrile (6**).**

7 (L-proline) $\xrightarrow[\text{reflux, 2 h}]{\text{CICH}_2\text{COCl, THF}}$ 8 (81%)

1. DCC, DCM, 15 °C • RT, 1 h

2. NH_4HCO_2 , RT, 1 h

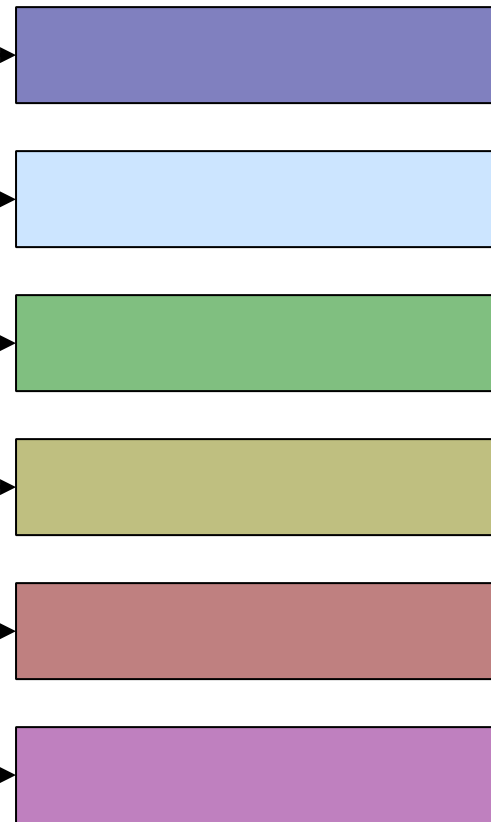
9 (82,5%)

1. TFAA, THF, 0 °C • RT, 2 h

2. NH_4HCO_2 , 5 °C • RT, toluene

6 (83%)

Page 2 of 5
Copyright in this article for the author(s) and the publisher(s).



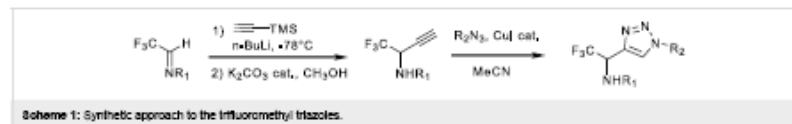
Textual output



More Problematic Sources

Detour

- Problematic documents:
 - Compound documents (pictures + text)
 - Complex layouts (magazines, journals)
 - Complex markup (math + chemical formulas)
- Already a lot of problems!



The copper(I)-catalyzed 1,3-dipolar cycloaddition [33-38] of organic azides and alkynes (also called “click chemistry”) resulting in the formation of 1,2,3-triazoles has become an increasingly attractive area [39]. According to the literature [33-38], the Cu(I) species can be used directly (e.g. CuI), or generated by oxidation of a Cu(0) or reduction of a Cu(II) species. Catalysis by the CuI is known to yield exclusively the 1,4-disubstituted regioisomer [33,34]. First, the *N*-(*p*-methoxyphenyl)-1-(trifluoromethyl)propargylamine was reacted with benzyl azide in the presence of CuI (10 mol%) and showed good reactivity with completion of the reaction within 24 h, whereas the use of CuSO₄/Na ascorbate afforded the cycloadduct in low yield. The reaction was then carried out with different propargylamines (*N*-(*p*-methoxyphenyl) and *N*-benzyl) and various azides at room temperature in acetonitrile within 24 h which afforded the compounds 2a-i with good yields (53-92%) after purification by column chromatography. The results are summarized in Table 1.

Table 1: Copper(I)-catalyzed synthesis of 1,4-disubstituted triazoles

Entry	R ₁	R ₂	Product	Yield (%) ^a
1	-FMP ^b	-Bn	2a	82
2	-FMP	-CH ₂ COPh	2b	76
3	-FMP	-CH ₂ COOC(CH ₃) ₃	2c	73
4	-FMP	-CH ₂ CO ₂ CH ₃	2d	83
5	-FMP	-CH ₂ CH ₂ OH	2e	87
6	-Bn ^a	-Bn	2f	75
7	-Bn	-CH ₂ COPh	2g	63
8	-Bn	-CH ₂ CO ₂ CH ₃	2h	92
9	-Bn	-CH ₂ CH ₂ OH	2i	73

^aFMP: *p*-methoxyphenyl, Bn: benzyl. ^bYield after flash purification.

As expected the new triazoles were formed in a fully regioselective manner affording the 1,4-regioisomer as highlighted from NOE experiments on compound 2c (Figure 1). A strong correlation was observed between the hydrogen H_a and H_b respectively. The structure of the other compounds 2a-i was assigned by analogy with 2c.

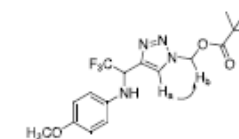
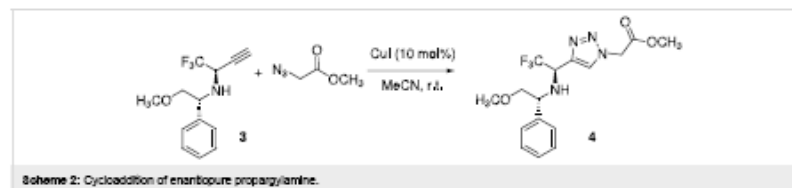


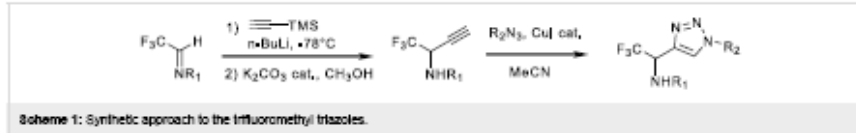
Figure 1: Experimentally found NOE correlation for compound 2c.

In our goal to study the influence of the CF₃ group on the conformation of peptidomimetics, we applied our strategy to the enantiopure trifluoromethyl-propargylamine 3 bearing the removable (*R*)-phenylglycinol chiral auxiliary (Scheme 2) [30-32].

good yield (79%) and as a single isomer without any racemization. This compound can easily afford the free amine ester which is a promising trifluoromethyl building block for the synthesis of new triazole-based trifluoromethyl oligomers.

The reaction was carried out under the same condition with azidoacetic acid methyl ester and afforded the cycloadduct 4 in





Long chemical entity names (row span)

The copper(I)-catalyzed 1,3-dipolar cycloaddition [33-38] of organic azides and alkynes (also called “click chemistry”) resulting in the formation of 1,2,3-triazoles has become an increasingly attractive area [39]. According to the literature [33-38], the Cu(I) species can be used directly (e.g. CuI), or generated by oxidation of a Cu(0) or reduction of a Cu(II) species. Catalysis by the CuI is known to yield exclusively the 1,4-disubstituted regioisomer [33,34]. First, the **N-(p-methoxyphenyl)-1-(trifluoromethyl)propargylamine** was reacted with benzyl azide in the presence of CuI (10 mol%) and showed good reactivity with completion of the reaction within 24 h, whereas the use of CuSO₄/Na ascorbate afforded the cycloadduct in low yield. The reaction was then carried out with different propargylamines (N-(p-methoxyphenyl) and N-benzyl) and various azides at room temperature in acetonitrile within 24 h which afforded the compounds 2a-i with good yields (63-92%) after purification by column chromatography. The results are summarized in Table 1.

References to entities mentioned in the table

As expected the new triazoles were formed in a fully regioselective manner affording the 1,4-regioisomer as highlighted (Figure 1). A strong hydrogen H_a and H_b compounds **2a-i** was

The reaction was carried out under the same condition with azidoacetic acid methyl ester and afforded the cycloadduct 4 in

“Basic” reaction scheme

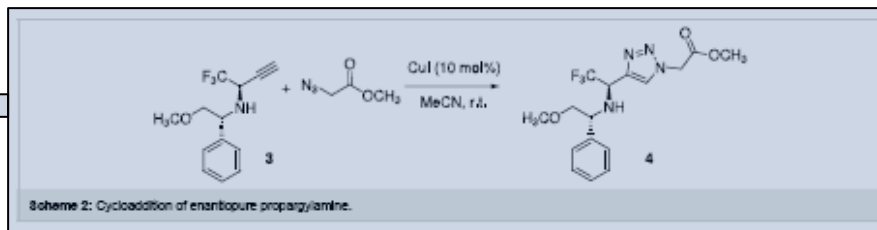


Table 1: Copper(I)-catalyzed synthesis of 1,4-disubstituted triazoles

Entry	R ₁	R ₂	Product	Yield (%) ^b
1	-FMP ^a	-Bn	2a	82
2	-FMP	-CH ₂ CO ₂ Ph	2b	76
3	-FMP	-CH ₂ OCCC(CH ₃) ₃	2c	73
4	-FMP	-CH ₂ CO ₂ CH ₃	2d	83
5	-FMP	-CH ₂ CH ₂ OH	2e	87
6	-Bn ^a	-Bn	2f	75
7	-Bn	-CH ₂ CO ₂ Ph	2g	63
8	-Bn	-CH ₂ CO ₂ CH ₃	2h	92
9	-Bn	-(CH ₂) ₂ OH	2i	73

^aFMP: p-methoxyphenyl, Bn: benzyl. ^bYield after flash purification.

Table with figure of a reaction

2a-i: Correspond to entities explained elsewhere

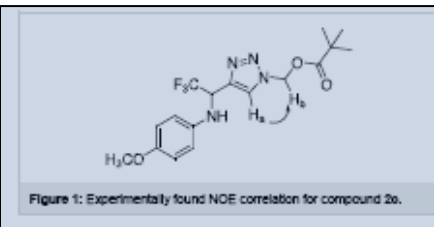


Figure with a reference to one special step in the table above (2c)

good yield (79%) and as a single isomer without any racemization. This compound can easily afford the free amine ester which is a promising trifluoromethyl building block for the synthesis of new triazole-based trifluoromethyl oligomers.



3. Suppose there are two events, x and y , in question with m possibilities for the first and n for the second. Let $p(i, j)$ be the probability of the joint occurrence of i for the first and j for the second. The entropy of the joint event is

$$H(x, y) = - \sum_{i,j} p(i, j) \log p(i, j)$$

while

$$H(x) = - \sum_{i,j} p(i, j) \log \sum_j p(i, j)$$

$$H(y) = - \sum_{i,j} p(i, j) \log \sum_i p(i, j).$$

It is easily shown that

3. Suppose there are two events, x and y , in question with m possibilities for the first and n for the second. Let p_{ij} be the probability of the joint occurrence of i for the first and j for the second. The entropy of the joint event is $H(x, y) = - \sum_{i,j} p_{ij} \log p_{ij}$

while

$H(x) = - \sum_{i,j} p_{ij} \log \sum_j p_{ij}$ $H(y) = - \sum_{i,j} p_{ij} \log \sum_i p_{ij}$

It is easily shown that

It is easily shown that

It is easily shown that



The Ultimate Horror

Detour

- Scanned or photographed documents with a rich structure
 - ... in bad quality
- Can only be processed with OCR software
 - Usually, a lot of errors



928 | Forschungsarbeiten

Chemie Ingenieur Technik 2008, 80, No. 7

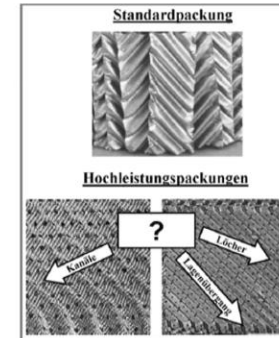


Abbildung 1. Strukturen verschiedener Standard- und Hochleistungspackungen.

Sollen sich durch Ungleichverteilungen der Phasen oder Ablagerungen in einzelnen Kanälen Strömungspässe ergeben, so ist mit einem frühzeitigen Fluten in diesen Zonen zu rechnen.

Bei schäumenden Systemen wurde festgestellt, dass die Fluidodynamik gelochter Packungsstrukturen ungünstig wirkt, da der Gasdurchtritt durch die Löcher blasen- und damit schaumfördernd wirkt.

Frage 2: Wirken gelochte Kanalstrukturen positiv auf die Leistungsfähigkeit von Packungen?
Im Betriebszustand strömt die Flüssigkeit in Form eines Films über die Packungsstruktur nach unten, während das Gas im verbleibenden Lückenvolumen aufwärts strömt. Einige Packungshersteller sehen gelochte Packungsbleche vor, um den Phasen die Möglichkeit zu geben, sich durch die Löcher der Bleche auszutauschen.

Aufgrund von Brückenbildungen überdeckt jedoch die Flüssigkeit oftmals die kleinen Löcher im Blech, sodass der Austausch der Strömungsphasen zwischen zwei benachbarten Kanälen eingeschränkt wird. Bei schäumenden Systemen wurde festgestellt, dass die Fluidodynamik gelochter Packungsstrukturen ungünstig wirkt, da der Gasdurchtritt durch die Löcher blasen- und damit schaumfördernd wirkt.

Bei Packungen mit niedrigen spezifischen Oberflächen ($\leq 300 \text{ m}^2/\text{m}^3$) ließe sich kein nennenswerter Einfluss der Löcher auf die Fluidodynamik von Packungen und deren Trennleistungsverhalten feststellen.

Frage 3: Ist die senkrechte Profilanstellung im Übergangsbereich der Packungslagen bei Hochleistungspackungen die fluidodynamisch effektivste Ausführungsform?

Bei den Kanalstrukturen von Standardpackungen läuft eine Kapazitätssteigerung zwangsläufig auf die senkrechte Anstellung der Strömungskanäle in den Übergangsbereichen der Packungslagen hinaus. Es lässt sich jedoch fragen, ob nicht durch eine gänzlich andere Packungsstruktur der Phasenübergang zwischen den Lagen erleichtert werden kann.

Abb. 2 (oben) zeigt die Struktur, die sich aus den beschriebenen Überlegungen und strömungstechnischen Untersuchungen ergeben hat. Die Struktur der Raschig Super-Pak wird durch schmale, wellenartige Schwünge gebildet, die alternierend aus dem Packungsblech nach oben und nach unten heraustreten. Die Schwünge verlaufen geneigt zur Horizontalen und die Bleche werden um 180° verdreht zueinander aufgereiht. Infolge der schmalen alternierenden Schwünge erhält die Raschig Super-Pak eine extrem offene Struktur, die keine Strömungspässe oder Fluterscheinungen sind damit ausgeschlossen. Des Weiteren findet sich in Abb. 2 (unten) auch die Ansicht der neuen Packungsstruktur in Strömungsrichtung des Gases beim Eintritt in eine neue Packungslage. Die extrem offene Struktur zwingt dem Gas beim Übertritt zur nächsten

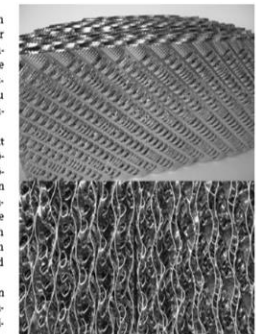


Abbildung 2. Raschig Super-Pak mit innovativer Lamellenstruktur (oben) und Ansicht einer Packungslage in Strömungsrichtung der Gasphase (unten).

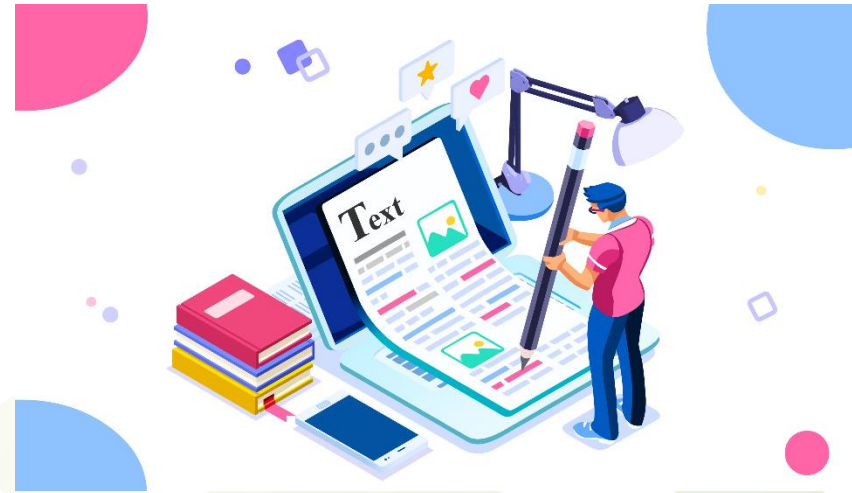
www.cit-journal.de

© 2008 Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim



Today's Lecture: Indexing

1. Document Preparation
- 2. Index Construction**
3. Query Evaluation





Index Construction

- **Building an inverted index looks easy:**
 1. Assign an ID to each document: **docID**
 2. Run the **document preparation** process on each document
 3. Compile a list of all **index terms**
 4. Assign an ID to each index term: **termID**
 5. Create a list of all **(termID, docID, tf) triplets**
 6. **Sort** this list: Primarily by termID, secondarily by docID
- Essentially, this corresponds to a **matrix transposition**
- Let's have a look at an example...



Example

- Our **example collection** of six documents:

1. The old night keeper keeps the keep in the town
2. In the big old house in the big old gown
3. The house in the town had the big old keep
4. Where the old night keeper never did sleep
5. The night keeper keeps the keep in the night
6. And keeps in the dark and sleeps in the light

- Case-folding, stopping, and stemming reduces the vocabulary to **ten index terms**:

- | | | | |
|---------|----------|----------|----------|
| 1. big | 4. house | 7. night | 10. town |
| 2. dark | 5. keep | 8. old | |
| 3. gown | 6. light | 9. sleep | |



Example

tf	1	2	3	4	5	6	7	8	9	10
1					3		1	1		1
2	2		1	1				2		
3	1			1	1			1		1
4					1		1	1	1	
5					3		2			
6		1			1	1			1	

- Then, we get the following **(termID, docID, tf)** triplets:

(5, 1, 3), (7, 1, 1), (8, 1, 1), (10, 1, 1), (1, 2, 2), (3, 2, 1), (4, 2, 1),
(8, 2, 2), (1, 3, 1), (4, 3, 1), (5, 3, 1), (8, 3, 1), (10, 3, 1), (5, 4, 1),
(7, 4, 1), (8, 4, 1), (9, 4, 1), (5, 5, 3), (7, 5, 2), (2, 6, 1), (5, 6, 1),
(6, 6, 1), (9, 6, 1)



Example

(5, 1, 3), (7, 1, 1), (8, 1, 1), (10, 1, 1), (1, 2, 2), (3, 2, 1), (4, 2, 1),
(8, 2, 2), (1, 3, 1), (4, 3, 1), (5, 3, 1), (8, 3, 1), (10, 3, 1), (5, 4, 1),
(7, 4, 1), (8, 4, 1), (9, 4, 1), (5, 5, 3), (7, 5, 2), (2, 6, 1), (5, 6, 1),
(6, 6, 1), (9, 6, 1)

- Now, **sort**: Primarily by termID, secondarily by docID

(1, 2, 2), (1, 3, 1), (2, 6, 1), (3, 2, 1), (4, 2, 1), (4, 3, 1), (5, 1, 3),
(5, 3, 1), (5, 4, 1), (5, 5, 3), (5, 6, 1), (6, 6, 1), (7, 1, 1), (7, 4, 1),
(7, 5, 2), (8, 1, 1), (8, 2, 2), (8, 3, 1), (8, 4, 1), (9, 4, 1), (9, 6, 1),
(10, 1, 1), (10, 3, 1)



Example

(1, 2, 2), (1, 3, 1), (2, 6, 1), (3, 2, 1), (4, 2, 1), (4, 3, 1), (5, 1, 3),
(5, 3, 1), (5, 4, 1), (5, 5, 3), (5, 6, 1), (6, 6, 1), (7, 1, 1), (7, 4, 1),
(7, 5, 2), (8, 1, 1), (8, 2, 2), (8, 3, 1), (8, 4, 1), (9, 4, 1), (9, 6, 1),
(10, 1, 1), (10, 3, 1)

- **The inverted index:**

Term	Posting List
1: big	(2, 2), (3, 1)
2: dark	(6, 1)
3: gown	(2, 1)
4: house	(2, 1), (3, 1)
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)
6: light	(6, 1)
7: night	(1, 1), (4, 1), (5, 2)
8: old	(1, 1), (2, 2), (3, 1), (4, 1)
9: sleep	(4, 1), (6, 1)
10: town	(1, 1), (3, 1)



Large Collections

- **Building the inverted index isn't difficult** if the whole document collection fits in **main memory**
- Now, let's get serious:
Typical collections are very large...
- What can we do?
 - **Sort-based inversion:**
Use an external (i.e. disk-based) sorting algorithm that works on compressed disk blocks (for performance reasons)
 - **Merge-based inversion:**
Read and index documents in memory until a fixed capacity is exceeded; when memory is full, the index is flushed to disk and merged with the index already stored on disk



Merge-Based Inversion

Documents

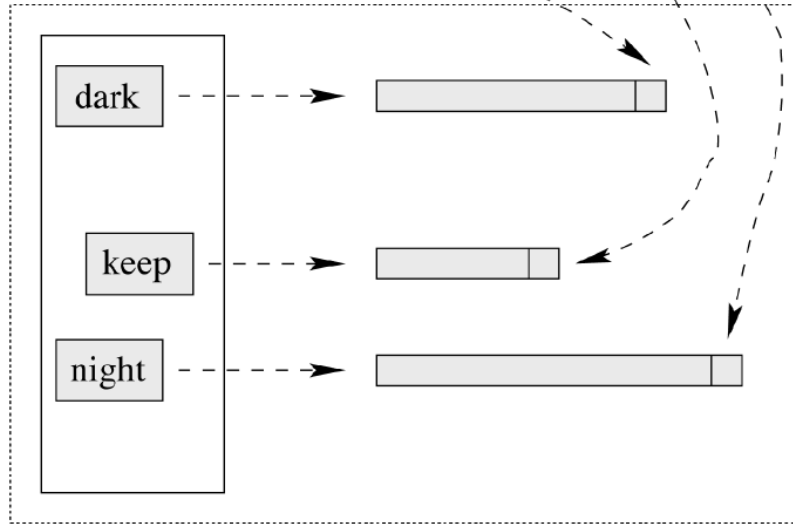


read

Parsed document

..., dark, ..., keep, night, ...

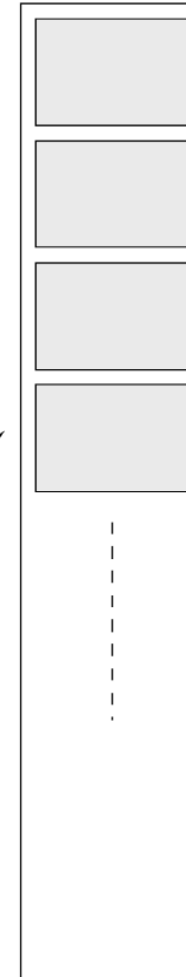
update



In-memory index

flush

Runs on disk



merge

Final index





Merge-Based Inversion

Merge-based indexing has several advantages:

- It is practical for collections of all sizes
- It even scales well and operates effectively in as little as 100 MB of main memory
- Disk space overheads can be restricted to a small fraction of the final index
- With clever data compression methods, the number of merge runs can be reduced further



Index Representations

- The problem of building the index essentially is solved
 - OK, Google uses massive replication and data distribution but these are very special requirements...
- Now, how to store an inverted index on disk?
- Since disk accesses are very expensive (e.g. compared to computations), there are **two major requirements**:

Keep the index as small as possible!

Read as little data as possible from disk!

- Since computational power comes at (almost) no cost, **effective data compression** is our first way to go!

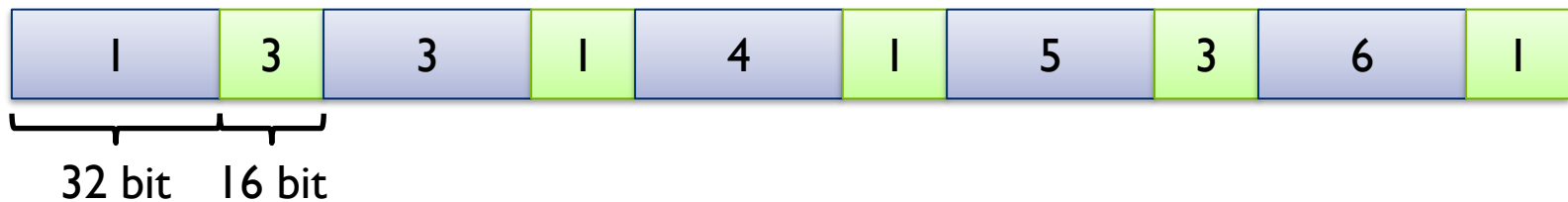


Index Representations

- A simple implementation of an inverted index:
 - Use 32-bit integers for document identifiers
 - Use 16-bit integers for term frequencies

Term	Posting List
	...
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)
	...

- Then, the **posting list for term “keep”** will be stored on disk like this:





Index Representations

- A typical inverted index:
 - Some document IDs occur very frequently in the whole index
 - Most document IDs occur very rarely
- Here, fixed-width integers are not space-efficient
- Furthermore, fixed-width integers limit the number of documents that can be stored...
- **Variable-length codes solve both problems**
 - They can encode **arbitrary large numbers**
 - They can be constructed to **store small values with little storage cost**, at the expense of large values
 - This perfectly fits our needs if document with many index entries get small IDs



Variable-Length Codes

- The simplest variable-bit infinite code is **unary**
 - Represent the integer $x > 0$ as $x - 1$ “1” bits followed by a terminating “0” bit
 - **Example:** Encode 12 by the bit sequence 11111111110
- Another variable-bit code is **Elias’ gamma code**:
 - To store the integer $x > 0$, it is factored into $2^a + b$, where $a = \lfloor \log_2(x) \rfloor$ and $0 \leq b < 2^a$
 - The codeword is formed as the concatenation of $a + 1$ represented in unary and b represented in fixed-width binary of width a
 - **Example:** Encode $12 = 2^3 + 4$ as 1110100



Variable-Length Codes

- Some more examples:

Value	Unary	Gamma
1	0	0
2	10	100
3	110	101
4	1110	11000
5	11110	11001
6	111110	11010
10	1111111110	1110010
100	...	1111110100100
1000	...	1111111110111101000



Variable-Length Codes

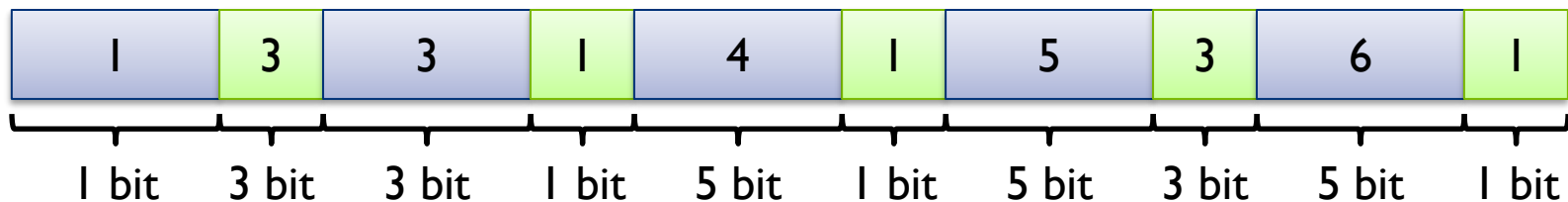
- The efficiency of each code depends on the distribution of input numbers to be encoded
- The unary code allows optimal space efficiency if the input distribution is given by $\Pr(x) = 2^{-x}$
 - Half of all values are the number 1
 - A quarter are 2
 - ...
- The gamma code is optimal for $\Pr(x) \approx 1 / (2x^2)$
- **Of course, there are many other codes available...**



Variable-Length Codes

Term	Posting List
	...
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)
	...

- Using gamma coding, our example posting list becomes:



- 28 bit** (compared to 240 using a fixed-length encoding)



Storing Gaps

Term	Posting List
	...
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)

+2 +1 +1 +1

- To allow even better compression ratios, we can **store gaps instead of document IDs**:

Term	Posting List (with Gaps)
	...
5: keep	(1, 3), (2, 1), (1, 1), (1, 3), (1, 1)
	...

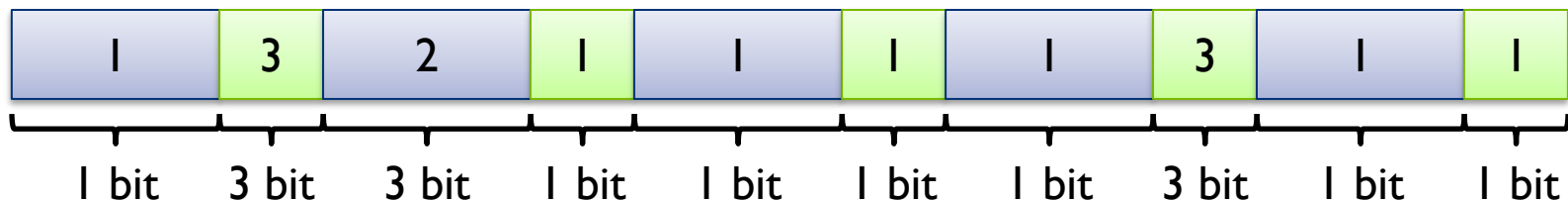
- Gaps usually are much smaller than document IDs...



Storing Gaps

Term	Posting List (with Gaps)
	...
5: keep	(1, 3), (2, 1), (1, 1), (1, 3), (1, 1)
	...

- What do we get?



- **16 bit** (compared to 28 and 240, respectively)
- Using gap storage, even large posting lists can be stored efficiently, which enables us to **abstain from stop words**



Skip Lists

- We already have seen that compression can help us a lot
- Now: How to reduce the number of disk accesses?

- As we have seen in Boolean retrieval, an important operation is **intersecting posting lists**
 - Inverted index:
 - step: {Document1, Document2}
 - mankind: {Document1}
 - China: {Document2}
 - Query:
 - “mankind AND step”
- **How to speed up this operation?**



Skip Lists

Term	Posting List (no Gaps)
	...
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)
6: light	(6, 1)
	...

- **Another example:**

When trying to answer the query “keep AND light,” we have to scan through the two posting lists shown above

- **Problem:** We have to scan the whole posting list of “keep” to finally reach document 6; we know that we can ignore every document having a smaller ID than 6
- **Is there any way to skip some of these postings?**

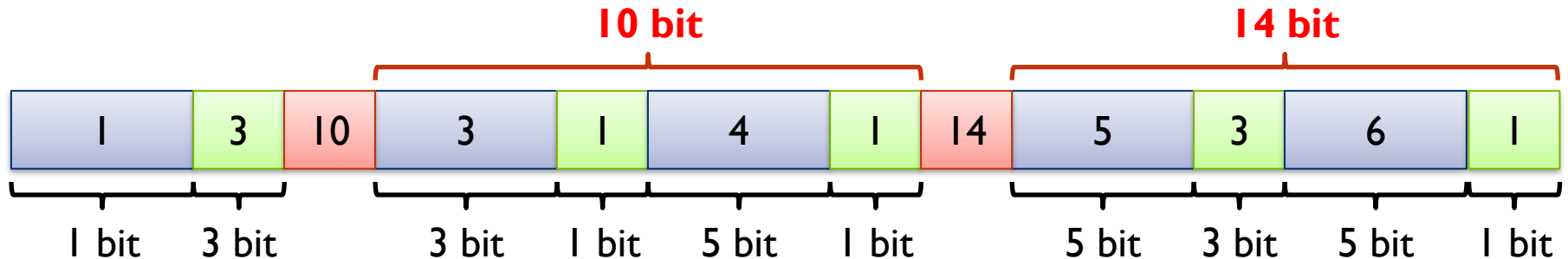


Skip Lists

- **Idea: Skip Lists**

- Evenly spaced, add some **skip pointers** to the list
- Every skip pointer consists of a number of bits that can be skipped to reach a different entry
- This skipped entries do not have to be accessed from disk

- Our gamma-coded posting list (no gaps) with skip pointers that allow skipping every other entry:

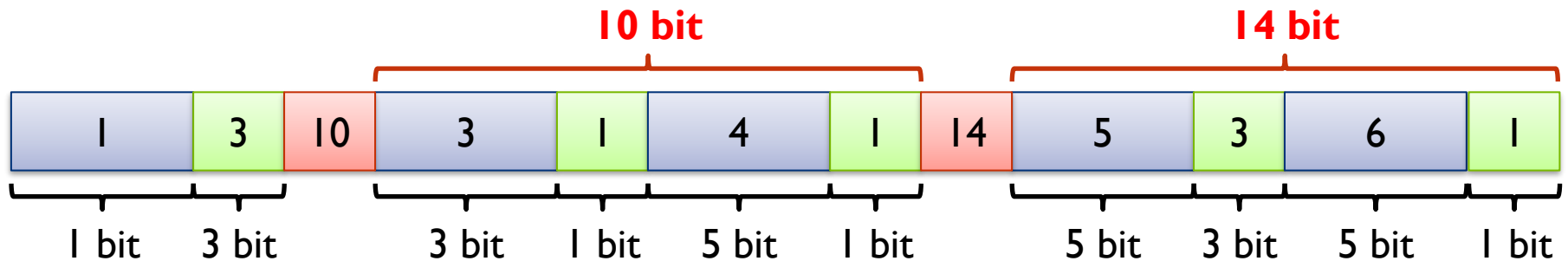


Note that in addition to this we need some coding mechanism to indicate whether an entry contains a skip pointer or not...



Skip Lists

- **Skip lists can speed up intersection operations massively if the posting lists are sorted by document ID**



- Where to place skip pointers?
 - A **heuristic** says that they should be placed **every \sqrt{k} postings** if k is the number of list entries...



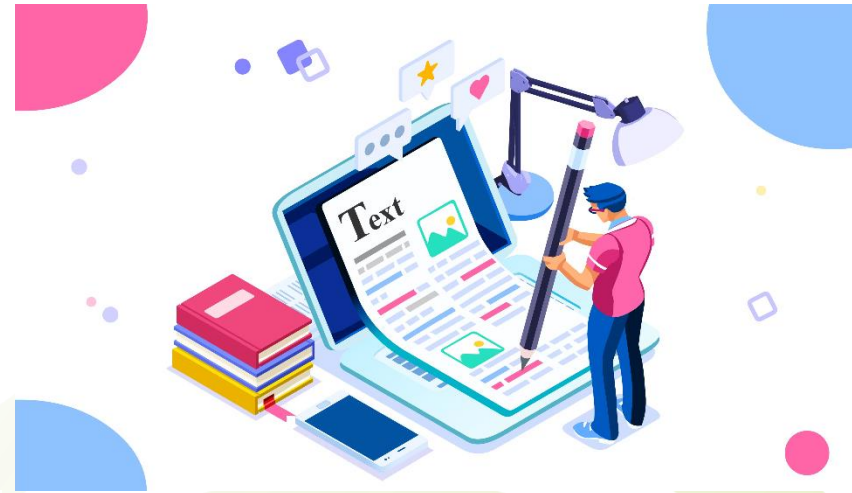
Dynamic Indexing

- How to handle changes to document collection?
 - New documents
 - Updated documents
 - Deleted documents
- **Simple (but inefficient) solution:**
Rebuild the index from scratch
- **Better solution:**
Keep an auxiliary in-memory index that keeps track of all changes
 - If the auxiliary index gets too large, it is merged with the main index



Today's Lecture: Indexing

1. Document Preparation
2. Index Construction
3. **Query Evaluation**





Query Processing

- **Boolean retrieval:**

Process queries as we already have discussed it

- **Vector space retrieval:**

Answer the query “dark, keep, night” by scanning through the postings lists for “dark,” “night,” and “keep,” while accumulating scores for each document

- Let’s have a look at an example...



Vector Space Retrieval

- **Query = “dark keep night”**
- Vector representation: Simple term frequencies
 - Query vector = (0, 1, 0, 0, 1, 0, 1, 0, 0, 0)

- Similarity measure:
Simple scalar product

Term	Posting List
2: dark	(6, 1)
5: keep	(1, 3), (3, 1), (4, 1), (5, 3), (6, 1)
7: night	(1, 1), (4, 1), (5, 2)

- Process query by scanning through the three lists and add up term frequencies for each occurring document
- This gives the following final scores:

Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6
3 + 1 = 4	0	1	1 + 1 = 2	3 + 2 = 5	1 + 1 = 2



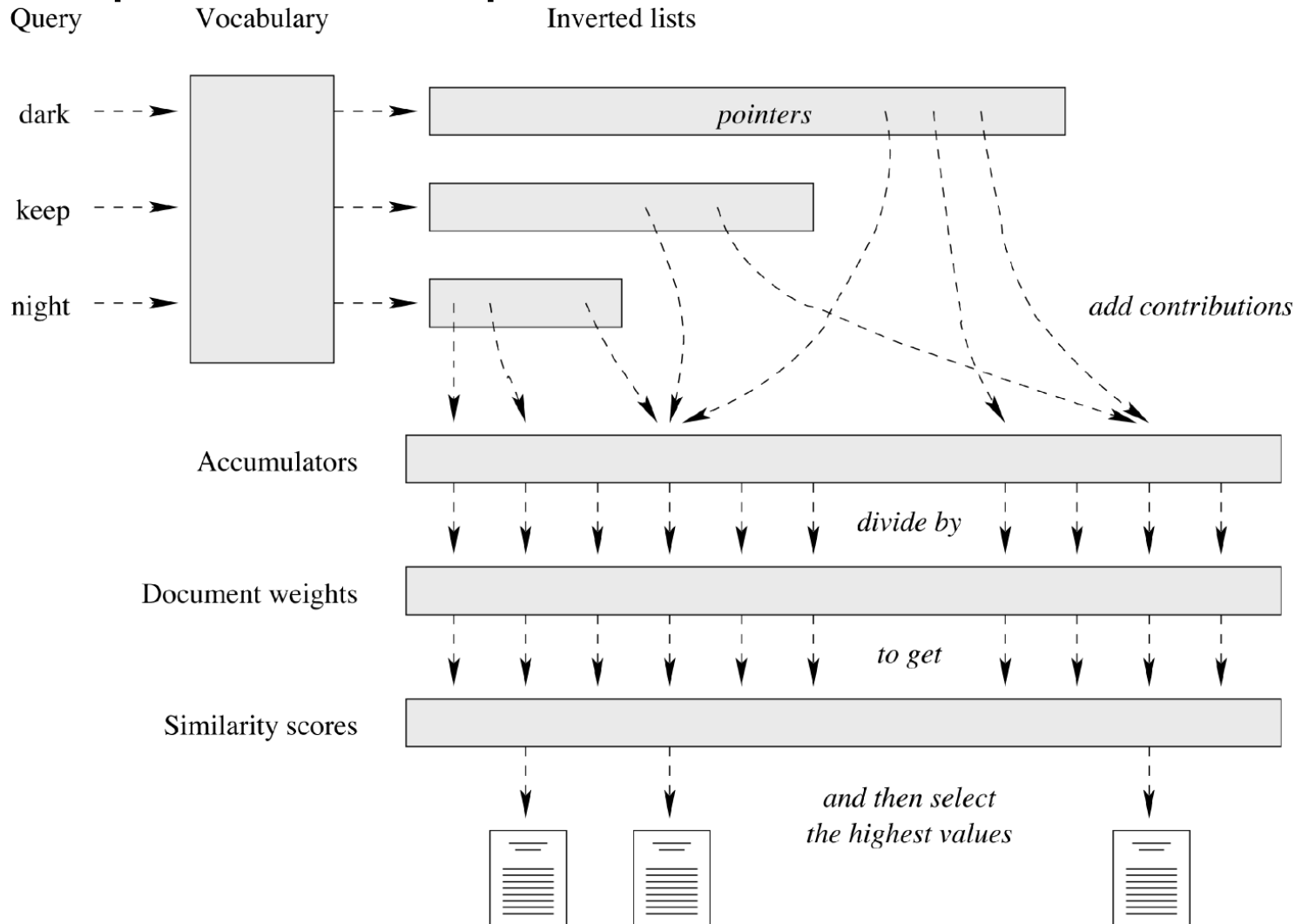
Vector Space Retrieval

- Due to the nature of the scalar product, we only need to **add up scores for any non-zero query component**
- To support more advanced vector representations, simply add some more information to the posting lists
 - For example, to support **TF-IDF**, store each term's IDF at the beginning of its corresponding posting list
- Furthermore, we can avoid reading all affected posting lists completely, by **sorting the postings by their TF**
 - This yields a significant speed-up of query processing
- **Similar approaches can be used to process queries for other retrieval models...**



Query Processing

The complete retrieval process:





Phrase Queries

- A special type of queries are **phrase queries**
- **Example:** “King of Finland”
- **Three strategies** to process phrase queries:
 - **Postprocessing:**
Initially, ignore the word order and do Boolean retrieval;
In a second step, search through the documents found and return only the ones containing the phrase
 - **Store word positions:**
Add word positions to each posting so that the locations of terms in documents can be checked during query evaluation
 - **Partial phrase indexes:**
Create a (partial) index containing phrases



Phrase Queries

- The three strategies complement each other and usually are applied in combination
 - Create a **phrase index** for phrases containing **frequent words** (phrases containing rare words can be found easily by using the other two approaches)
 - Store **word positions** for every word and phrase (can be done efficiently using compression)
 - If for some reason there are postings without word positions, postprocess all document finds by doing a phrase search



Next Lecture

- Latent Semantic Indexing

