Overview: Information Retrieval

• Information retrieval is about
  – Analyzing and indexing text to answer user queries
  – “Finding the needle in the haystack”
  – Being effective and efficient
  – Scalable Algorithms (Web Scale Data)
Overview: Data Mining

- Finding new & interesting information from data
  - Association rules
  - Clusterings
  - Classifiers

<table>
<thead>
<tr>
<th>Transaction id</th>
<th>milk</th>
<th>bread</th>
<th>beer</th>
<th>butter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Tentative Agenda/Content

Introduction

Ranking principles
Link analysis
Indexing & searching
Information extraction

Frequent itemsets & association rules
Unsupervised clustering
(Semi-)supervised classification
Advanced topics in data mining

Information Retrieval

Data Mining
People

• Lecturer:
  – Prof. Sebastian Michel
  – smichel (at) cs.uni-kl.de

• Teaching Assistant:
  – MSc. Koninika Pal
  – pal (at) cs.uni-kl.de

http://dbis.informatik.uni-kl.de/
Organization & Regulations

• **Lecture:**
  – Wednesday
  – 10:00 – 11:30
  – Room 42-110

• **Exercise:**
  – Tuesday (bi-weekly)
  – 15:30 - 17:00
  – Room 52-204
  – **First session: May 5**
Exercises

• **Assignment sheet, every two weeks**

• Mixture of:
  – Practical: E.g., apply algorithms on “paper”
  – Theory: Where appropriate (show that …)
  – Possibly, brief essay: Explain the difference of x and y (i.e., short summary)

• **Need to successfully participate to be admitted to final exam**

• Regulations on next slides
Regulations for Admission to Exam

• Successful participation in exercise sessions
• There will be 5 exercise sheets
• Each comprises 3 mandatory assignments
• No handing in of solutions, instead:
  – Tutor asks at beginning of TA session to mark on a sheet the assignments you have solved and can present

<table>
<thead>
<tr>
<th>Name</th>
<th>Assignment 1</th>
<th>Assignment 2</th>
<th>Assignment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Britney Clinton</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

....
Regulations for Admission to Exam (2)

• Each mark is equivalent to one point
• You **need to obtain 10 points** throughout the semester **to get admitted to the exam**
• **Full point** is given if solution is correct or close to it
• **Zero points** is given if assignment has proven incorrect to large extent
• **Zero points on entire sheet** will be given in case you marked an assignment solved but it is obvious you didn’t really do it (->cheating)
Exam

• Written or oral **exam at the end of teaching period in semester** (last week or week thereafter)

• Everything mentioned in lecture or exercises is relevant for exam. Unless explicitly stated.

• We assume you actively participated in the exercises to be prepared.
Note in Credit Points and Work

• Lecture is worth 4 ECTS points
• Each point is assumed to describe 30 hours of work
• 4 x 30h = 120h
• 14 weeks, makes around 9h of work each week
Registration

• If not done already, please register through the KIS system
  • Registration is closing on May 10, 2015
  • Without registration, no marks in TA session possible, hence, no exam qualification.
Literature Information Retrieval

Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze.
Introduction to Information Retrieval
Website: http://nlp.stanford.edu/IR-book/

Modern Information Retrieval: The concepts and technology behind search.
Addison-Wesley, 2010.
Website: http://www.mir2ed.org

W. Bruce Croft, Donald Metzler, Trevor Strohman.
Search Engines: Information Retrieval in Practice.
Addison-Wesley, 2009.
Website: http://www.pearsonhighered.com/croft1epreview/

Datamining Literature will be announced a bit later.
How would you Process a Query over Text Documents?

• Finding all **term occurrences**?
• How many time does a term need to be included? At least once (**boolean**)?
• Or do you **count term occurrences**?
• Is every term equally suitable to be used to assess a document?
• What about the url of the website? **Are some websites better than others?**
Information Retrieval in a Nutshell

- Web, intranet, digital libraries, desktop search
- Unstructured/semi-structured data

- **crawl**
  - handle dynamic pages, detect duplicates, detect spam

- **extract & clean**
  - strategies for crawl schedule and priority queue for crawl frontier

- **index**
  - build and analyze web graph, index all tokens or word stems

- **match**
  - fast top-k queries, query logging, auto-completion

- **rank**
  - scoring function over many data and context criteria

- **present**
  - GUI, user guidance, personalization

Server farms with 10,000's (2002) – 100,000's (2010) computers, distributed/replicated data in high-performance file system (GFS, HDFS,…), massive parallelism for query processing (MapReduce, Hadoop,…)

Information Retrieval and Data Mining, SoSe 2015, S. Michel
Promotion: IS-Project

• Every **winter semester**
• **Task: Implement a web search engine**
• 2014/15 for the first time
• **Directly related to THIS lecture!**
  - if you have participated in the IS-Project this lecture will appear very familiar to you
  - if you plan to participate in the IS-Project, this lecture (together with core DB content you all know) will give you the foundations to handle the project easily (depending also on Java Skills!)
Ergebnisse für: informatik kaiserslautern

Suche dauerte 3060ms

Chair of Software Engineering: Dependability:
#1 | Score: 100 | http://seda.cs.uni-kl.de/impressum/
... Adresse http://seda.informatik.uni-kl.de : Anschrift Leitung Lehrstuhl Software Engineering: Dependability Technische Universität Kaiserslautern Postfach 3049 ... Universit?t Kaiserslautern Gottlieb-Daimler-Stra?e, Geb?ude 47 67663 Kaiserslautern Univ.-Prof. Dr. rer. nat. 名誉工学博士 (湘南工科大学)* Helmut kaiserslautern

disco | distributed computer systems lab kaiserslautern
#2 | Score: 99.3 | http://disco.cs.uni-kl.de
... of Kaiserslautern Tweets by @DISCO_Research Tweets von @DISCO_TUKL Write your thesis with a disco advisor We offer a variety of bachelor and master theses at any point in the informatik

Chair of Software Engineering: Dependability:
#3 | Score: 96.3 | http://seda.cs.uni-kl.de/staff/liggesmeyer/
... of Kaiserslautern Building 32, Room 425 P.O. Box 3049 67653 Kaiserslautern Germany Telephone: +49 (631) 205-3328 Secretary: +49 (631) 205-3341 (Caroline Frey) Fax: +49 (631) 205-3331 E-mail: informatik

Software Engineering: Process Measurement Research Group:
#4 | Score: 96.0 | http://wwwwanse.cs.uni-kl.de/impressum/
Promotion: IS-Project (3)

Promotion: IS-Project (4)

1: http://www.cs.uni-kl.de/aktuelles/

2: http://cs.uni-kl.de/
   ...Prof. Liggesmeyer im Interview mit dem Deutschlandfunk zum Wissenschaftsjahr 2014 "Digitale Gesellschaft" Prof. Dr. Arnd Poetzsch-Heffter zum Vizepräsidenten für Forschung, Technologie und Innovation gewählt. -Feed der aktuellen Meldungen Schnellzugriff Fachbereich ...

3: http://cs.uni-kl.de/aktuelles/vizepraesident_ph/
   ...Poetzsch-Heffter herzlich zu dieser besonderen Auszeichnung. Weitere Informationen Homepage zur Homepage von Prof. Dr. Arnd Poetzsch-Heffter [mehr ...] Pressemeldung TU-KL Pressemeldung der TU-Kaiserslautern [mehr ...] ...

4: http://www.cs.uni-kl.de/
   ...Prof. Liggesmeyer im Interview mit dem Deutschlandfunk zum Wissenschaftsjahr 2014 "Digitale Gesellschaft" Prof. Dr. Arnd Poetzsch-Heffter zum Vizepräsidenten für Forschung, Technologie und Innovation gewählt. -Feed der aktuellen Meldungen Schnellzugriff Fachbereich ...

5: http://informatik.uni-kl.de/
   ...Prof. Liggesmeyer im Interview mit dem Deutschlandfunk zum Wissenschaftsjahr 2014 "Digitale Gesellschaft" Prof. Dr. Arnd Poetzsch-Heffter zum Vizepräsidenten für Forschung, Technologie und Innovation gewählt. -Feed der aktuellen Meldungen Schnellzugriff Fachbereich ...
Information Retrieval

Ranking Principles
Part I: Ranking Principles

I.1 Boolean Retrieval & Document Processing
   Boolean Retrieval, Tokenization, Stemming, Lemmatization

I.2 Basic Ranking & Evaluation Measures
   TF*IDF, Vector Space Model, Precision/Recall, F-Measure, etc.

I.3 Probabilistic Retrieval Models
   Probabilistic Ranking Principle,
   Binary Independence Model, BM25

I.4 Statistical Language Models
   Unigram Language Models, Smoothing,
   Extended Language Models

I.5 Latent Topic Models
   (Probabilistic) Latent Semantic Indexing,
   Latent Dirichlet Allocation

I.6 Advanced Query Types
   Relevance Feedback, Query Expansion, Novelty & Diversity
I.1 Boolean Retrieval & Document Processing

1. Definition of Information Retrieval
2. Boolean Retrieval
3. Document Processing
4. Spelling Correction and Edit Distances

Based on MRS Chapters 1 & 3
Shakespeare...

• Which plays of Shakespeare mention Brutus and Caesar but not Calpurnia?

(i) Get all of Shakespeare’s plays from [Project Gutenberg](https://www.gutenberg.org) in plain text

(ii) Use UNIX utility `grep` to determine files that match Brutus and Caesar but not Calpurnia

```
grep --files-with-matches 'Brutus' * | xargs grep --files-with-matches 'Caesar' | xargs grep --files-without-match 'Calpurnia'
```
1. Definition of Information Retrieval

Information retrieval is **finding material** (usually documents) of an **unstructured nature** (usually text) that satisfies an **information need** from within **large collections** (usually stored on computers).

- **Finding documents** (e.g., articles, web pages, e-mails, user profiles) as opposed to creating additional data (e.g., statistics)
- **Unstructured data** (e.g., text) w/o easy-for-computer structure as opposed to structured data (e.g., relational database)
- **Information need** of a user, usually expressed through a query, needs to be satisfied which implies effectiveness of methods
- **Large collections** (e.g., Web, e-mails, company documents) demand scalability & efficiency of methods
2. Boolean Retrieval Model

- **Boolean variables** indicate presence of words in documents
- **Boolean operators** AND, OR, and NOT
- Boolean queries are arbitrarily **complex compositions** of those
  - Brutus AND Caesar AND NOT Calpurnia
  - NOT ((Duncan AND Macbeth) OR (Capulet AND Montague))
  - ...
- **Query result is (unordered) set of documents satisfying the query**
Incidence Matrix

- Binary word-by-document matrix indicating presence of words
  - Each column is a binary vector: which document contains which words?
  - Each row is a binary vector: which word occurs in which documents?
  - To answer a Boolean query, take rows corresponding to query words and apply Boolean operators column-wise
Extended Boolean Retrieval Model

• Boolean retrieval **used to be the standard and is still common** in certain domains (e.g., library systems, patent search)

• **Plain Boolean queries are too restricted**
  – Queries look for words anywhere in the document
  – Words have to be exactly as specified in the query

• **Extensions of the Boolean retrieval model**
  – Proximity operators to demand that words occur close to each other (e.g., with at most k words or sentences between them)
  – Wildcards (e.g., Ital*) for a more flexible matching
  – Fields/Zones (e.g., title, abstract, body) for more fine-grained matching
  – ...
Boolean Ranking

• Boolean query can be satisfied by many zones of a document
• Results can be ranked based on how many zones satisfy query
  – Zones are given weights (that sum to 1)
  – Score is the sum of weights of those fields that satisfy the query
  – **Example**: Query Shakespeare in title, author, and body
    • Title with weight 0.3, author with weight 0.2, body with weight 0.5
    • Document that contains Shakespeare in title and body but not in author gets score 0.8
• **How to convert natural language documents into an easy-for-computer format?**

• Words can be simply misspelled or in various forms
  – plural/singular (e.g., car, cars, foot, feet, mouse, mice)
  – tense (e.g., go, went, say, said)
  – adjective/adverb (e.g., active, actively, rapid, rapidly)
  – ...

• Issues and solutions are often highly language-specific (e.g., diacritics and inflection in German, accents in French)

• Important first step in IR
What is a Document?

• If data is not in **linear plain-text** format (e.g., ASCII, UTF-8), it needs to be converted (e.g., from PDF, Word, HTML)

• **Data has to be divided into documents as retrievable units**
  – Should the book “Complete Works of Shakespeare” be considered a single document? Or, should each act of each play be a document?
  – UNIX mbox format stores all e-mails in a single file. Separate them?
  – Should one-page-per-section HTML pages be concatenated?
Tokenization

• **Tokenization** *splits a text into tokens*

  Two households, both alike in dignity, in fair Verona, where

  Two households both alike in dignity in fair Verona where

• A **type** is a class of all tokens with the same character sequence

• A **term** is a (possibly normalized) type that is included into an IR system’s dictionary and thus indexed by the system

• **Basic tokenization**
  (i) Remove punctuation (e.g., commas, fullstops)
  (ii) Split at white spaces (e.g., spaces, tabulators, newlines)
Issues with Tokenization

- Language- and content-dependent
  - Boys’ => Boys vs. can’t => can’t
  - http://www.cs.uni-kl.de and support@ebay.com
  - co-ordinates vs. good-looking man
  - straight forward, white space, Los Angeles
  - l’ensemble and un ensemble
  - Compounds:
    Lebensversicherungsgesellschaftsansgestellter
  - No spaces at all (e.g., major East Asian languages)
Stopwords

• Stopwords are **very frequent words that carry no information** and are thus excluded from the system’s dictionary (e.g., *a, the, and, are, as, be, by, for, from*)

• Can be defined explicitly (e.g., with a list) or implicitly (e.g., as the k most frequent terms in the collection)

• **Do not seem to help with ranking documents**

• Removing them saves significant space but **can cause problems**
  – to be or not to be, the who, etc.
  – “president of the united states”, “with or without you”, etc.

• Current trend towards shorter or no stopword lists
Stemming

• **Variations of words could be grouped together** (e.g., plurals, adverbial forms, verb tenses)
• A crude heuristic is to cut the ends of words (e.g., ponies => poni, individual => individu)
• Word stem is not necessarily a proper word
• **Variations of the same word ideally map to same unique stem**
• Popular stemming algorithms for English
  – Porter ([http://tartarus.org/martin/PorterStemmer/](http://tartarus.org/martin/PorterStemmer/))
  – Krovetz
• For English stemming has little impact on retrieval effectiveness
Stemming: Example

Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth the fatal loins of these two foes

Two household, both alik in digniti,
In fair Verona, where we lay our scene,
From ancient grudg break to new mutini,
Where civil blood make civil hand unclean.
From forth the fatal loin of these two foe
Lemmatization

• Lemmatizer conducts full morphological analysis of the word to identify the lemma (i.e., dictionary form) of the word

• Example: For the word saw, a stemmer may return s or saw, whereas a lemmatizer tries to find out whether the word is a noun (return saw) or a verb (return to see)

• For English lemmatization does not achieve considerable improvements over stemming in terms of retrieval effectiveness
Other Ideas

• **Diacritics** (e.g., ü, ø, à, œ)
  – Remove/normalize diacritics: ü => u, å => a, ø => o
  – Queries often do not include diacritics (e.g., les miserables)
  – Diacritics are sometimes typed using multiple characters: für => fuer

• **Lower/upper-casing**
  – Discard case information (e.g., United States => united states)

• **n-grams as sequences of n characters** (inter- or intra-word) are useful for Asian (CJK) languages without clear word spaces
What’s the Effect?

• *Depends on the language; effect is typically limited with English*

• Results for 8 European languages [Hollink et al. 2004]
  – **Diacritic** removal helped with Finnish, French, and Swedish
  – **Stemming** helped with Finnish (30% improvement) but only little with English (0-5% improvement and even less with lemmatization)
  – **Compound splitting** helped with Swedish (25%) and German (5%)
  – **Intra-word 4-grams** helped with Finnish (32%), Swedish (27%), and German (20%)

• Larger benefits for morphologically richer languages
Zipf’s Law (after George Kingsley Zipf)

- The collection frequency $cf_i$ of the $i$-th most frequent word in the document collection is inversely proportional to the rank $i$

$$cf_i \propto \frac{1}{i}$$

- For the relative collection frequency with language-specific constant $c$ (for English $c \approx 0.1$) we obtain

$$\frac{cf_i}{\sum_j cf_j} = \frac{c}{i}$$

- In an English document collection, we can thus expect the most frequent word to account for $10\%$ of all term occurrences.
Zipf’s Law (cont’d)

\[ L_D = \sum_j c f_j \]
Heaps’ Law (after Harold S. Heaps)

- The **number of distinct words** $|V|$ in a document collection (i.e., the size of the vocabulary) relates to the **total number of word occurrences** as

$$|V| = k \left( \sum_{v \in V} cf(v) \right)^b$$

with collection-specific constants $k$ and $b$

- We can thus expect the size of the vocabulary to grow with the size of the document collection
Heaps’ Law (cont’d)

![Graph showing Heaps' Law with k = 35, b = 0.5 and the New York Times dataset.](image)

- Heaps’ Law formula: $L_D = k V^{1/b}$
- $L_D$: Length of the document
- $V$: Number of distinct words
- $k$: Heaps' constant
- $b$: Exponent that determines the slope of the log-log plot
4. Spelling Correction and Edit Distances

• Often, users don’t know how to spell.
• When user types in unknown, potentially misspelled word, we can try to map it to “closest” term in dictionary
• Need notion of distance between terms
  – adding extra character (e.g., hoouse vs. house)
  – omitting character (e.g., huse)
  – using wrong character (e.g., hiuse)
  – as-heard spelling (e.g., kwia vs. choir)
Hamming Edit Distance

• Distances should satisfy triangle inequality
  \[ d(x, z) \leq d(x, y) + d(y, z) \] for strings \( x, y, z \) and distance \( d \)

• **Hamming edit distance is the number of positions at which the two strings \( x \) and \( y \) are different**

• Strings of different lengths compared by **padding shorter** one with null characters (e.g., house vs. hot => house vs. hot _ _ )

• **Hamming edit distance counts wrong characters**

• **Examples:**
  \[ d(\text{car}, \text{cat}) = 1 \]
  \[ d(\text{house}, \text{hot}) = 3 \]
  \[ d(\text{house}, \text{hoouse}) = 4 \]
Longest Common Subsequence

• A **subsequence** of two strings $x$ and $y$ is a string $s$ such that all characters from $s$ occur in $x$ and $y$ in the same order but not necessarily contiguously.

• **Longest common subsequence (LCS)** distance defined as

$$d(x, y) = \max(|x|, |y|) - \max_{s \in S(x,y)} |s|$$

with $S(x, y)$ as the set of all subsequences of $x$ and $y$ and string lengths $|x|$, $|y|$, and $|s|$.

• LCS distance counts omitted characters.

• **Examples:**
  - $d(\text{house}, \text{huse}) = 1$
  - $d(\text{banana}, \text{atana}) = 2$
Levenshtein Edit Distance

• **Levenshtein** edit distance between two strings \( x \) and \( y \) is the minimal number of **edit operations** (insert, replace, delete) required to transform \( x \) into \( y \)

• The **minimal number of operations** \( m[i, j] \) to transform the prefix substring \( x[1:i] \) into \( y[1:j] \) is defined via

\[
m[i, j] = \min \begin{cases} 
m[i-1, j-1] + (x[i] = y[j] ? 0 : 1) & \text{(replace } x[i]?) \\
m[i-1, j] + 1 & \text{(delete } x[i]) \\
m[i, j-1] + 1 & \text{(insert } y[j]) 
\end{cases}
\]

• **Examples:**
  
  – \( d(\text{hoouse, house}) = 1 \)
  – \( d(\text{house, rose}) = 2 \)
  – \( d(\text{house, hot}) = 3 \)

- Edit operations can be weighted (e.g., based on letter frequency)
- Or different “penalty” for replace, delete, insert.
Levenshtein Edit Distance (cont’d)

• Levenshtein edit distance between two strings \(x\) and \(y\) corresponds to \(m[|x|, |y|]\) and can be computed using **dynamic programming** in time and space \(O(|x| \cdot |y|)\)
Levenshtein Edit Distance: Example
Dynamic Programming

- **Example:** cat vs. kate

What choices do we have?
- **replace:** replacing “c” with “k” and add cost of 1 to cost of m[0,0], i.e., distance of empty strings?
- **delete:** delete “c” and add 1 to cost of m[0,1], i.e., distance of empty string to “k”?
- **insert:** Insert “k” and add 1 to cost of m[1,0], i.e., distance of string “c” to empty string

<table>
<thead>
<tr>
<th></th>
<th>k</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$m[i, j] = \min \begin{cases} 
 m[i - 1, j - 1] + (x[i] = y[j] ? 0 : 1) & \text{(replace } x[i]?) \\
 m[i - 1, j] + 1 & \text{(delete } x[i]) \\
 m[i, j - 1] + 1 & \text{(insert } y[j]) 
\end{cases}$
Levenshtein Edit Distance: Example Dynamic Programming (2)

• **Example:** cat vs. kate

```
  -  k  a  t  e
-  0  1  2  3  4
 c  1  1   ?
 a  2   3   
 t  3   
```

What choices do we have?

• **replace:** replacing “c” with “a” => add cost of 1 to cost of m[0,1], i.e., distance of empty string with “k”
• **delete:** delete “c” => add 1 to cost of m[0,2], i.e., distance of empty string to “ka”?
• **insert:** insert “a” => add 1 to cost of m[1,1], i.e., distance of string “c” to “k”

\[
m[i, j] = \min \begin{cases} 
m[i - 1, j - 1] + (x[i] = y[j] \ ? \ 0 : 1) & (\text{replace } x[i]) \\
m[i - 1, j] + 1 & (\text{delete } x[i]) \\
m[i, j - 1] + 1 & (\text{insert } y[j]) 
\end{cases}
\]
**Levenshtein Edit Distance: Example Dynamic Programming (3)**

- **Example:** cat vs. kate

Now looking at (1,3), i.e., distance of string “c” to “kat”

\[
m[i, j] = \min \begin{cases} 
m[i-1, j-1] + (x[i] = y[j] \ ? 0 : 1) & \text{(replace } x[i]\text{)} 
m[i-1, j] + 1 & \text{(delete } x[i]\text{)} 
m[i, j-1] + 1 & \text{(insert } y[j]\text{)} 
\end{cases}
\]
**Levenshtein Edit Distance: Example**

**Dynamic Programming (4)**

**Example:** cat vs. kate

![Alignment chart](image)

$$m[i, j] = \min \begin{cases} 
m[i-1, j-1] + (x[i] = y[j] \land 0 : 1) & \text{(replace } x[i]?) \\
m[i-1, j] + 1 & \text{(delete } x[i]) \\
m[i, j-1] + 1 & \text{(insert } y[j]) 
\end{cases}$$
Summary of I.1

• **Boolean retrieval**
supports precise-yet-limited querying of document collections

• **Stemming and lemmatization**
to deal with syntactic diversity (e.g., inflection, plural/singular)

• **Zipf’s law**
about the frequency distribution of terms

• **Heaps’ law**
about the number of distinct terms

• **Edit distances**
to handle spelling errors and allow for vagueness
Additional Literature for I.1

• V. Holliink, J. Kamps, C. Monz, and M. de Rijke: Monolingual document retrieval for European languages, IR 7(1):33-52, 2004

• A. Singhal: Challenges in running a commercial search engine, SIGIR 2005