Compact Indexes for Flexible Top-k Retrieval

Simon Gog [1] Matthias Petri[2]

[1]Institute of Theoretical Informatics,
Karlsruhe Institute of Technology

[2]Computing and Information Systems,
The University of Melbourne

July 1th 2015

Top-k document retrieval

Given

- Collection $\mathcal{D}' = \{d_1, \ldots, d_{N-1}\}$
- Each d_i is a string over alphabet $\Sigma' = [2, \sigma]$ terminated by sentinel character 1 (also #)
- $\mathcal{D} = \mathcal{D}' \cup d_N$, with $d_N = 0$.
- „Bag of words" query $Q = \{q_0, q_1, \ldots, q_{m-1}\}$ (unordered set of size m)

Problem

Given a collection $\mathcal{D}$, a query Q of length m, and a similarity measure $\mathcal{S} : \mathcal{D} \times \mathcal{P}_{=m}(\Sigma') \to \mathbb{R}$. Calculate the top-$k$ documents of $\mathcal{D}$ with regard to Q and $\mathcal{S}$. That is a sorted list of document identifiers $\mathrm{T} = \{\tau_0, \ldots, \tau_{k-1}\}$, with $\mathcal{S}(d_{\tau_i}, Q) \geq \mathcal{S}(d_{\tau_{i+1}}, Q)$ for $0 \leq i < k$ and $\mathcal{S}(d_{\tau_{k-1}}, Q) \geq \mathcal{S}(d_j, Q)$ for $j \notin \mathrm{T}$.

Example

Fix a concatenation $\mathcal{C}$ of $\mathcal{D}$.

$i =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathcal{C}^{word} =$	LA	O	LA	#	O	LA	LA	LA	#	O	O	LA	#	$
$\mathcal{C} =$	2	3	2	1	3	2	2	2	1	3	3	2	1	0
	d_0				d_1					d_2				d_3

- $\mathcal{S}^{sfreq}(d, q) := f_{d,q}$ (i.e. single-term frequency ranking)
- $\mathcal{S}^{sfreq}(d_0, LA) = 2$,
  $\mathcal{S}^{sfreq}(d_1, LA) = 3$,
  $\mathcal{S}^{sfreq}(d_2, LA) = 1$,
  $\mathcal{S}^{sfreq}(d_3, LA) = 0$.
- Top-2: $\mathrm{T} = \{1, 0\}$

Previous and related work

- Optimal time ($O(|q_0| + k)$) and space solution for single-term frequency ranking by Navarro & Nekreich (SODA'12)
- Multi-term ranking for term frequency using linear space and time dependent on $n^{1-\frac{1}{m}}$ by Hon et al. (J. ACM 2014)
- Larson et al. (CPM'14): Reduction of boolean matrix multiplication to problem of finding elements which contain both terms of a two-term query

Previous and related work

- Optimal time ($O(|q_0| + k)$) and space solution for single-term frequency ranking by Navarro & Nekreich (SODA'12)
- Multi-term ranking for term frequency using linear space and time dependent on $n^{1-\frac{1}{m}}$ by Hon et al. (J. ACM 2014)
- Larson et al. (CPM'14): Reduction of boolean matrix multiplication to problem of finding elements which contain both terms of a two-term query

Our goal

A practical solution for multi-term queries and a wide range of similarity measures, like...

Okapi BM25 similarity measure

Successful IR similarity measure:

$$S_{Q,d}^{\mathrm{BM25}} = \sum_{q \in Q} \underbrace{\frac{(k_1+1) f_{d,q}}{k_1 \left(1 - b + b \frac{n_d}{n_{\mathrm{avg}}}\right) + f_{d,q}}}_{=w_{d,q}} \cdot \underbrace{f_{Q,q} \cdot \ln\left(\frac{N - F_{\mathcal{D},q} + 0.5}{F_{\mathcal{D},q} + 0.5}\right)}_{=w_{Q,q}}$$

depends on 3 document-dependent factors:

- $f_{d,q}$ term frequency (# of occurrences of term q in d)
- $F_{\mathcal{D},q}$ document frequency (# of distinct ds which contain q)
- n_d length of document d

Okapi BM25 similarity measure

Successful IR similarity measure:

$$S_{Q,d}^{\mathrm{BM25}} = \sum_{q \in Q} \underbrace{\frac{(k_1+1) f_{d,q}}{k_1 \left(1 - b + b \frac{n_d}{n_{\mathrm{avg}}}\right) + f_{d,q}}}_{=w_{d,q}} \cdot \underbrace{f_{Q,q} \cdot \ln\left(\frac{N - F_{\mathcal{D},q} + 0.5}{F_{\mathcal{D},q} + 0.5}\right)}_{=w_{Q,q}}$$

depends on 3 document-dependent factors:

- $f_{d,q}$ term frequency (# of occurrences of term q in d)
- $F_{\mathcal{D},q}$ document frequency (# of distinct ds which contain q)
- n_d length of document d

Okapi BM25 similarity measure

Successful IR similarity measure:

$$S_{Q,d}^{\mathrm{BM25}} = \sum_{q \in Q} \underbrace{\frac{(k_1+1) f_{d,q}}{k_1 \left(1 - b + b \frac{n_d}{n_{\mathrm{avg}}}\right) + f_{d,q}}}_{=w_{d,q}} \cdot \underbrace{f_{Q,q} \cdot \ln\left(\frac{N - F_{\mathcal{D},q} + 0.5}{F_{\mathcal{D},q} + 0.5}\right)}_{=w_{Q,q}}$$

depends on 3 document-dependent factors:

- $f_{d,q}$ term frequency (# of occurrences of term q in d)
- $F_{\mathcal{D},q}$ document frequency (# of distinct ds which contain q)
- n_d length of document d

Okapi BM25 similarity measure

Successful IR similarity measure:

$$S_{Q,d}^{\mathrm{BM25}} = \sum_{q \in Q} \underbrace{\frac{(k_1+1) f_{d,q}}{k_1 \left(1 - b + b \frac{n_d}{n_{\mathrm{avg}}}\right) + f_{d,q}}}_{=w_{d,q}} \cdot \underbrace{f_{Q,q} \cdot \ln\left(\frac{N - F_{\mathcal{D},q} + 0.5}{F_{\mathcal{D},q} + 0.5}\right)}_{=w_{Q,q}}$$

depends on 3 document-dependent factors:

- $f_{d,q}$ term frequency (# of occurrences of term q in d)
- $F_{\mathcal{D},q}$ document frequency (# of distinct ds which contain q)
- n_d length of document d

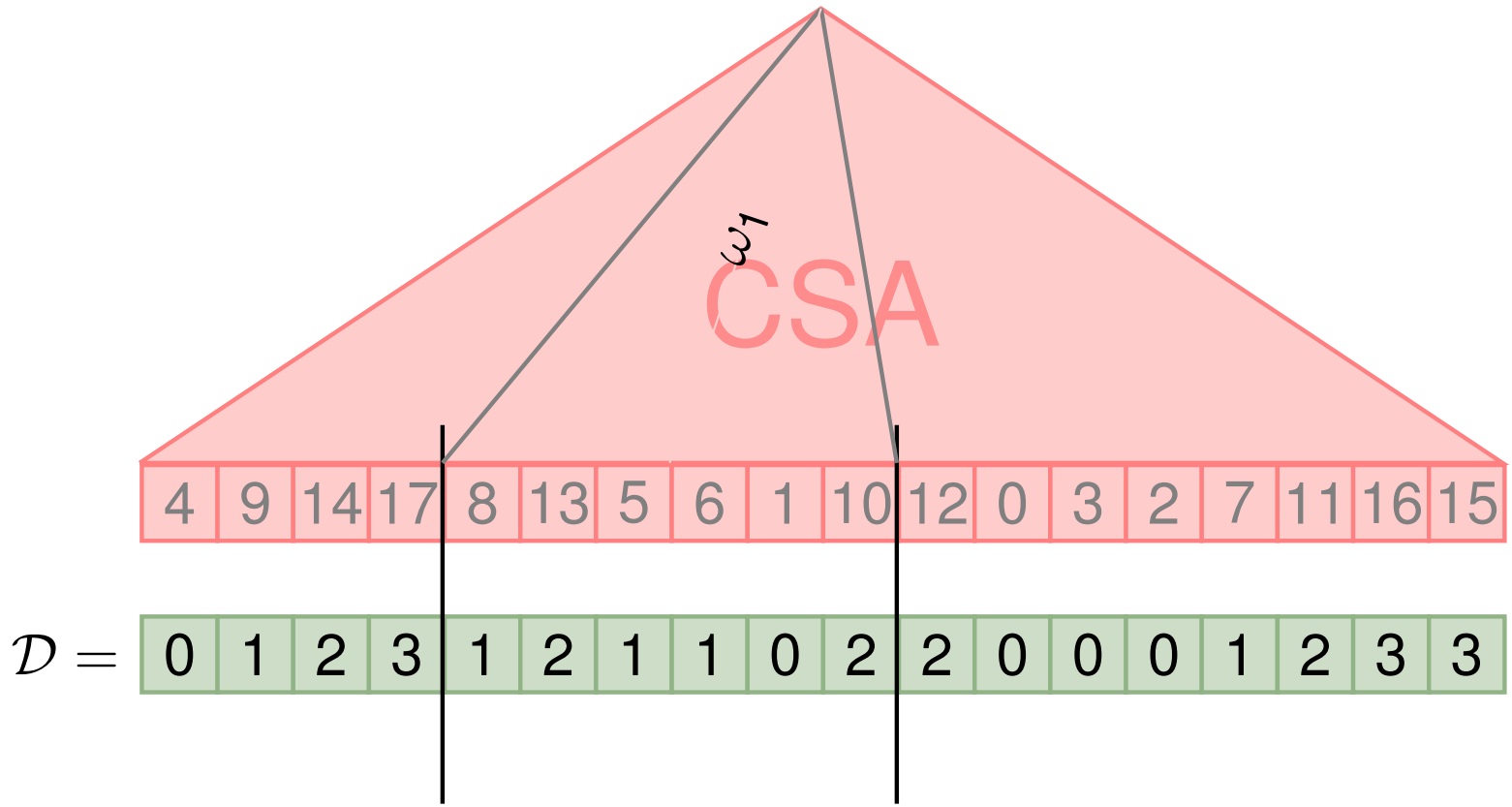
Previous practical solution

The GREEDY framework of Culpepper et al. (ESA'10) [1] consists of

- a Compressed Suffix Array (CSA) of concatenation $\mathcal{D}$
- Wavelet Tree of the Document Array of $\mathcal{D}$
- CSA provides phrase search and snippet extraction
- This functionality is missing in Inverted Indexes (II)

The GREEDY framework

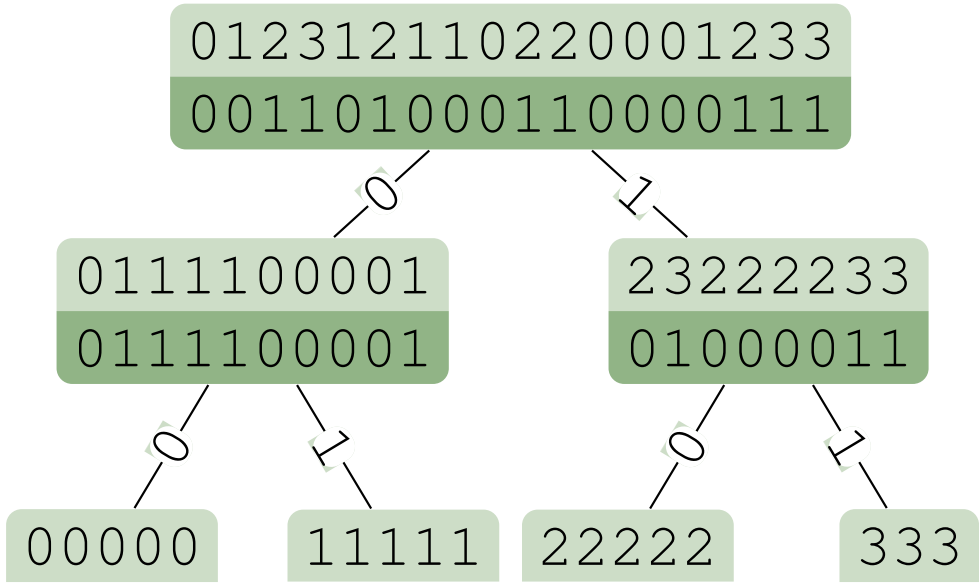
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\mathcal{T} =$	ω_2	ω_1	ω_3	ω_3	#	ω_1	ω_1	ω_4	ω_1	#	ω_1	ω_4	ω_3	ω_1	#	ω_5	ω_5	#
$b =$	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1



Interval of $q = \omega_1$ in $\mathcal{D}$ corresponds to the (multi)set of documents which contain q.

The GREEDY framework

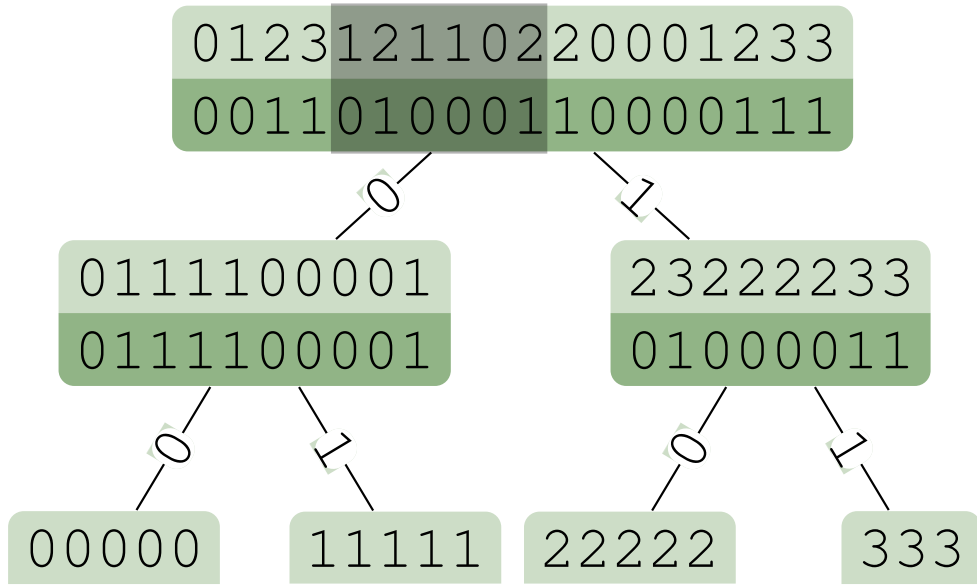
- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)



Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

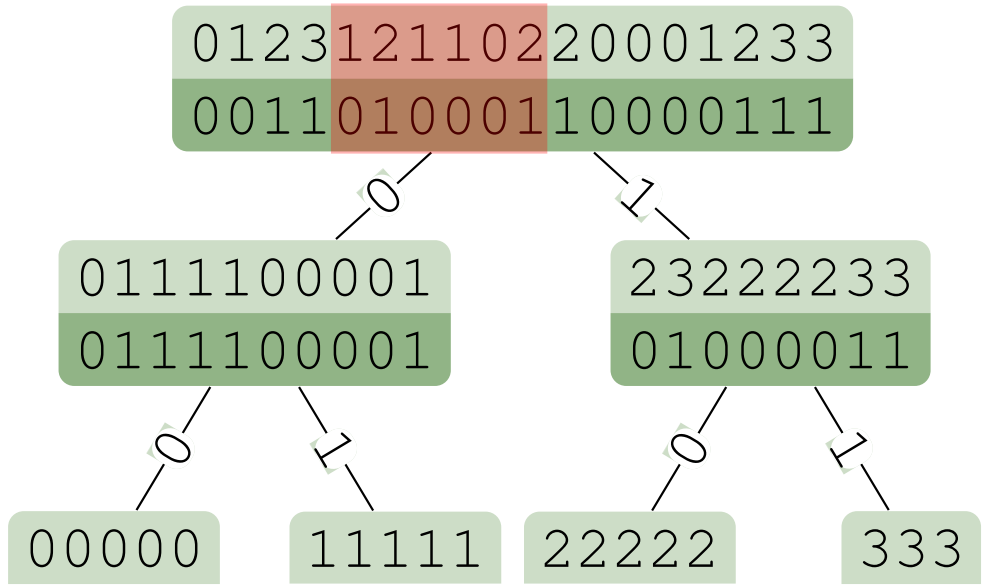


push state

Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

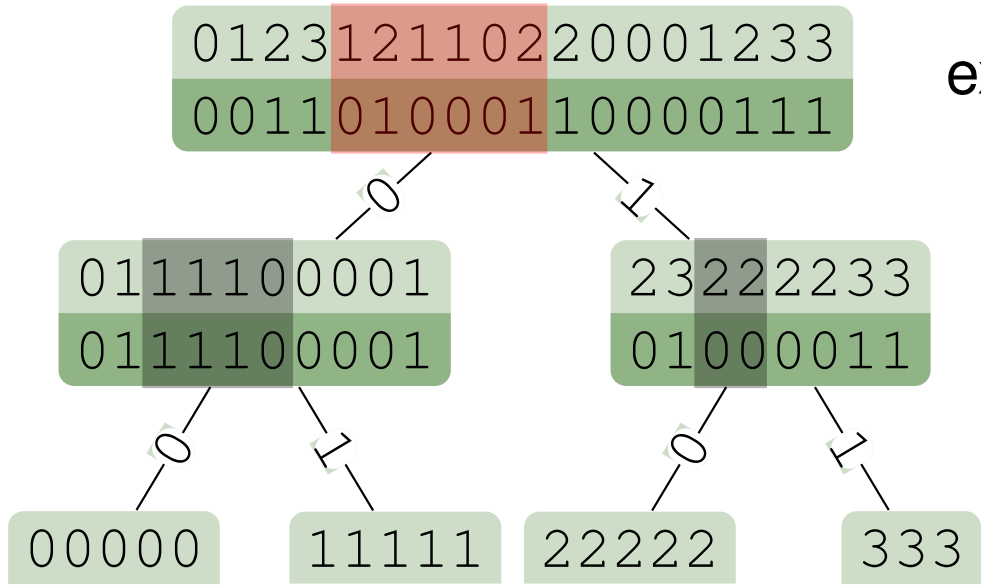


pop state

Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

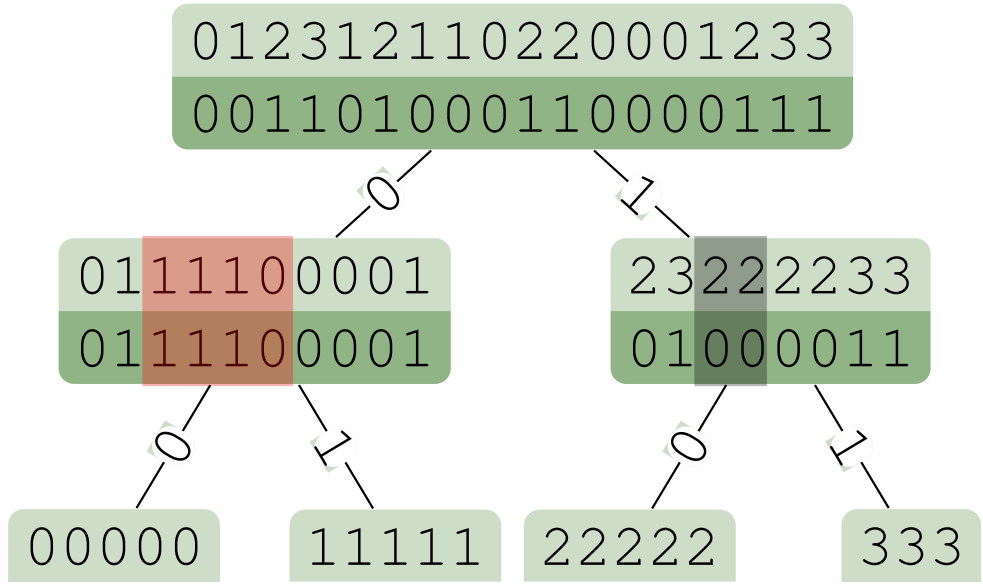


expand ($\mathcal{O}(1)$ time) and push

Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

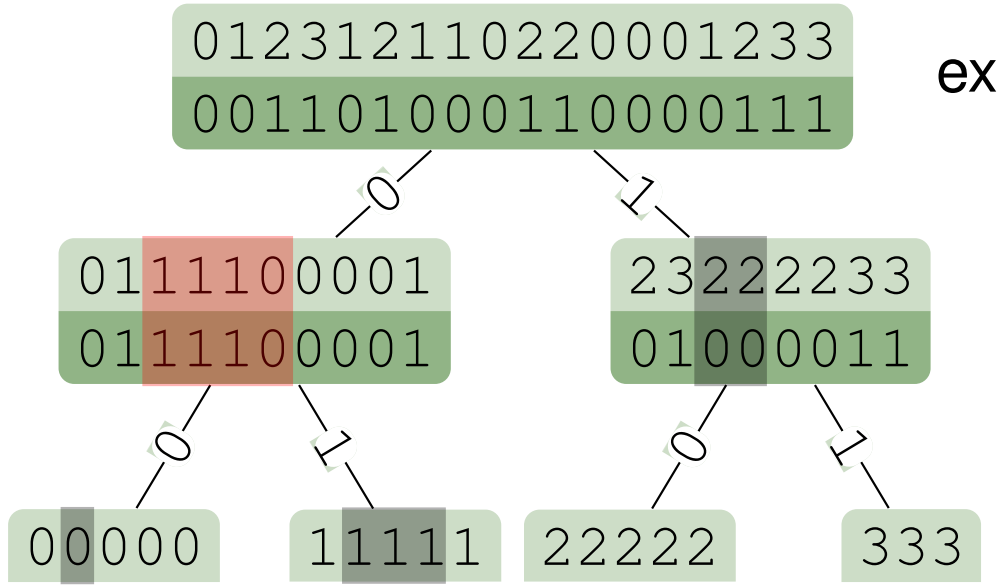


pop state

Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

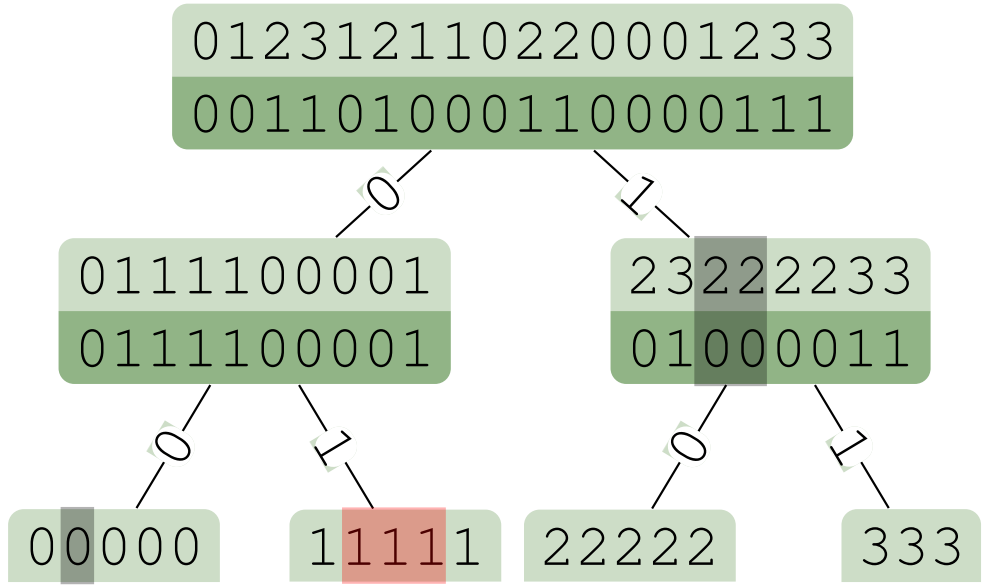


expand and push

Top documents containing ω_1:

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)

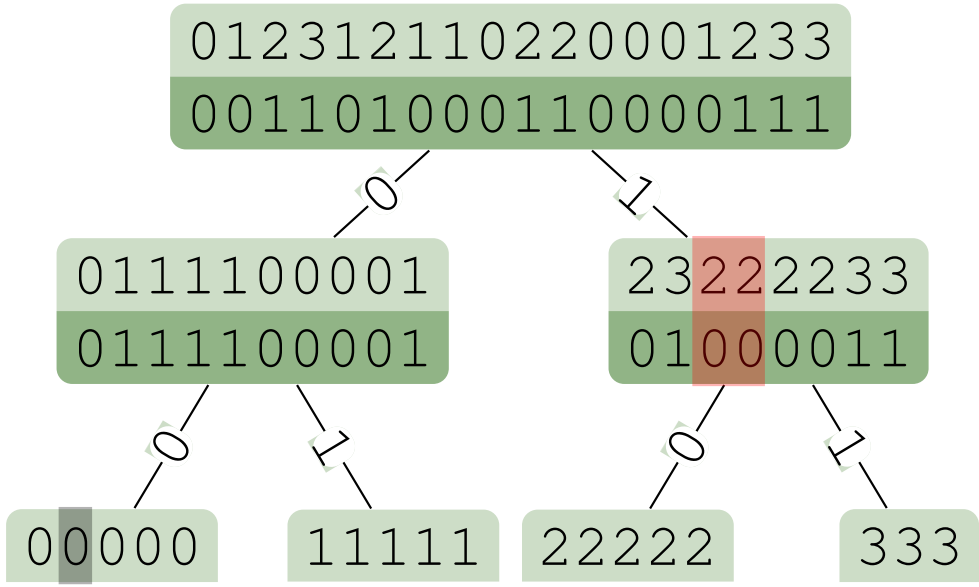


pop and report

Top documents containing ω_1: d_1 (3 times)

The GREEDY framework

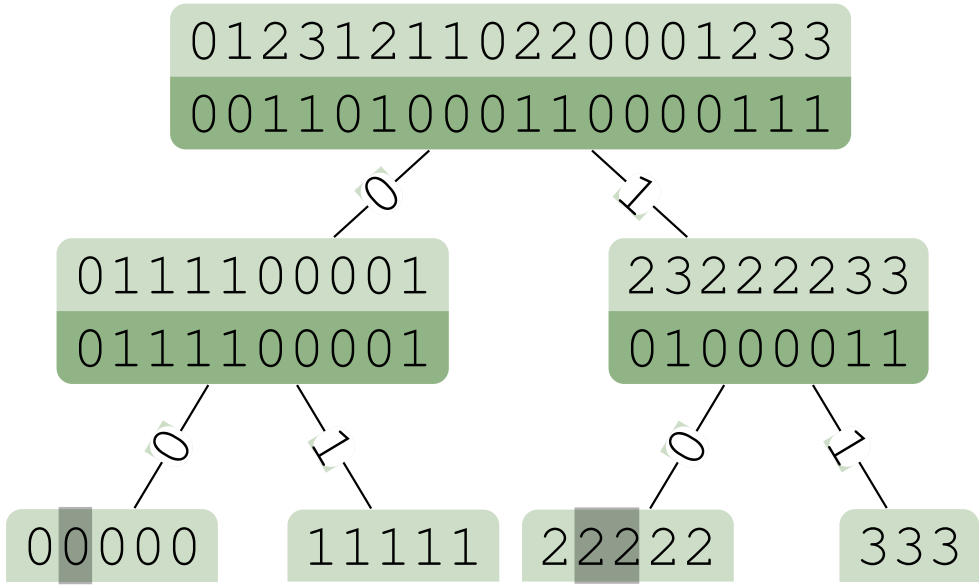
- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)



Top documents containing ω_1: d_1 (3 times)

The GREEDY framework

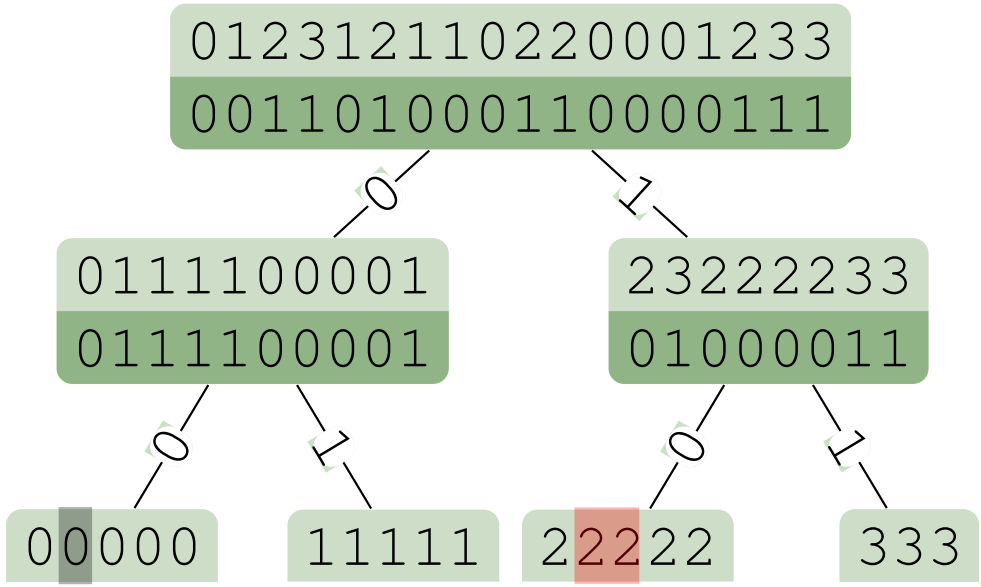
- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)



Top documents containing ω_1: d_1 (3 times)

The GREEDY framework

- Represent document array $\mathcal{D}$ as wavelet tree *WTD*
- Example: Search top-2 documents (frequency based ranking)



Top documents containing ω_1: d_1 (3 times), d_2 (2 times)

The GREEDY framework (conclusion)

- Elegant and simple alogrithm
- Size: $|CSA(\mathcal{T})| + |WT(\mathcal{D})|$
- $= nH_k(\mathcal{T}) + n \log |D| + o(n \log |D|)$ using a plain WT
- Worst case time depends on # distinct docs in lex. range
- Culpepper et al. (ESA'10) showed that it is practical for single-term frequency ranking, however
  - large index size (character-based indexing)
  - only for small collections

Generalize and improve GREEDY

- multi-term (state consists of multiple intervals)
- ranked-and or ranked-or version
- more complex similarity measures: TF$\times$IDF, BM25, LMDS
- Implemenation: 64-bit, word-alphabet

Three tricks to achieve a better score estimation:

- overestimate $\max_{d \in \mathcal{D}_v}\{f_{d,q}\}$:
  interval size - unqiue docs in interval + 1
- Use repetition array to get unique docs in $\mathcal{O}(1)$
- $\min_{d \in \mathcal{D}_v}\{n_d\}$ can be determined in $\mathcal{O}(1)$ time (sort doc IDs according to length)

Generalize and improve GREEDY

- multi-term (state consists of multiple intervals)
- ranked-and or ranked-or version
- more complex similarity measures: TF$\times$IDF, BM25, LMDS
- Implemenation: 64-bit, word-alphabet

Three tricks to achieve a better score estimation:

- overestimate $\max_{d \in \mathcal{D}_v}\{f_{d,q}\}$:
  interval size - unqiue docs in interval + 1
- Use repetition array to get unique docs in $\mathcal{O}(1)$
- $\min_{d \in \mathcal{D}_v}\{n_d\}$ can be determined in $\mathcal{O}(1)$ time (sort doc IDs according to length)

Generalize and improve GREEDY

- multi-term (state consists of multiple intervals)
- ranked-and or ranked-or version
- more complex similarity measures: TF$\times$IDF, BM25, LMDS
- Implemenation: 64-bit, word-alphabet

Three tricks to achieve a better score estimation:

- overestimate $\max_{d \in \mathcal{D}_v}\{f_{d,q}\}$:
  interval size - unqiue docs in interval + 1
- Use repetition array to get unique docs in $\mathcal{O}(1)$
- $\min_{d \in \mathcal{D}_v}\{n_d\}$ can be determined in $\mathcal{O}(1)$ time (sort doc IDs according to length)

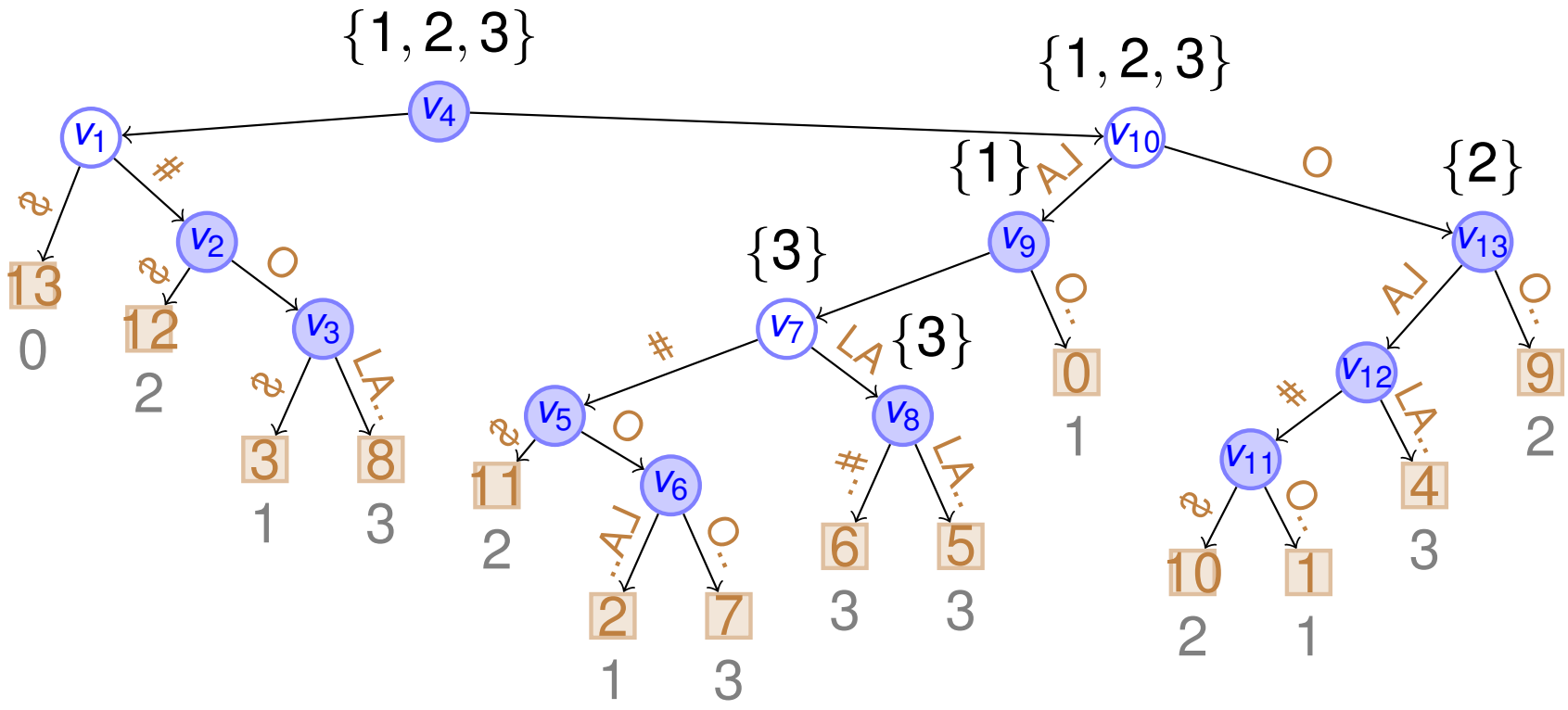
Generalize and improve GREEDY

- multi-term (state consists of multiple intervals)
- ranked-and or ranked-or version
- more complex similarity measures: TF$\times$IDF, BM25, LMDS
- Implemenation: 64-bit, word-alphabet

Three tricks to achieve a better score estimation:

- overestimate $\max_{d \in \mathcal{D}_v}\{f_{d,q}\}$:
  interval size - unqiue docs in interval + 1
- Use repetition array to get unique docs in $\mathcal{O}(1)$
- $\min_{d \in \mathcal{D}_v}\{n_d\}$ can be determined in $\mathcal{O}(1)$ time (sort doc IDs according to length)

Document Frequency: Binary Suffix Tree (BST)



Document Frequency

Sadakane's [2] solution.

- H is at most $2n$ bits
- add $o(n)$-bit select structure

For $[l, r] \leftarrow CSA.search(q)$:

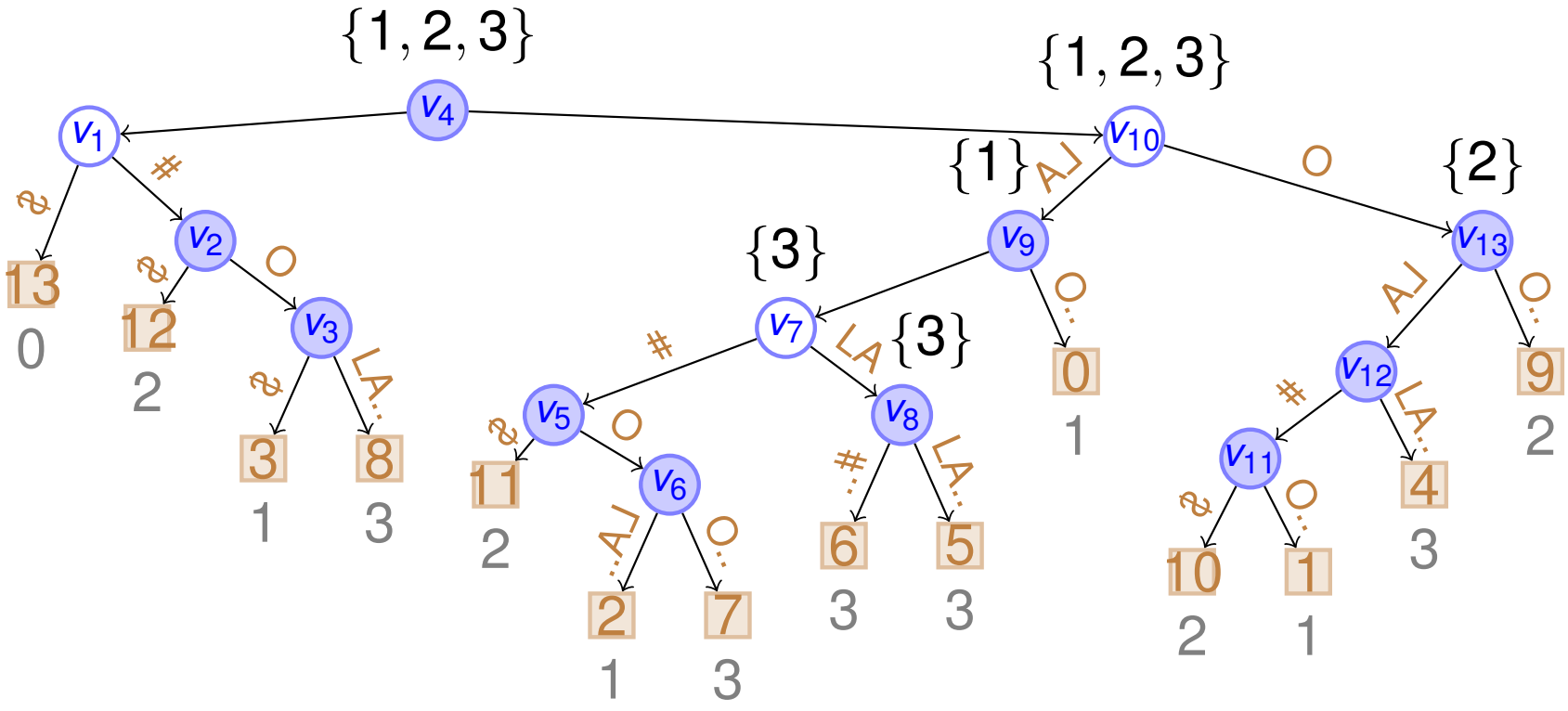
```
00  document_frequency(H, [l, r])
01      s ← r − l + 1
02      y ← select(H, r, 1)
03      if l = 0 then
04          return s − (y − r + 1)
05      else
06          x ← select(H, l, 1)
07          return s − (y − r + 1 − (x − l + 1))
```

Document Frequency for Subsets $\mathcal{D}_v$

Solution for document subsets represented in *WTD*

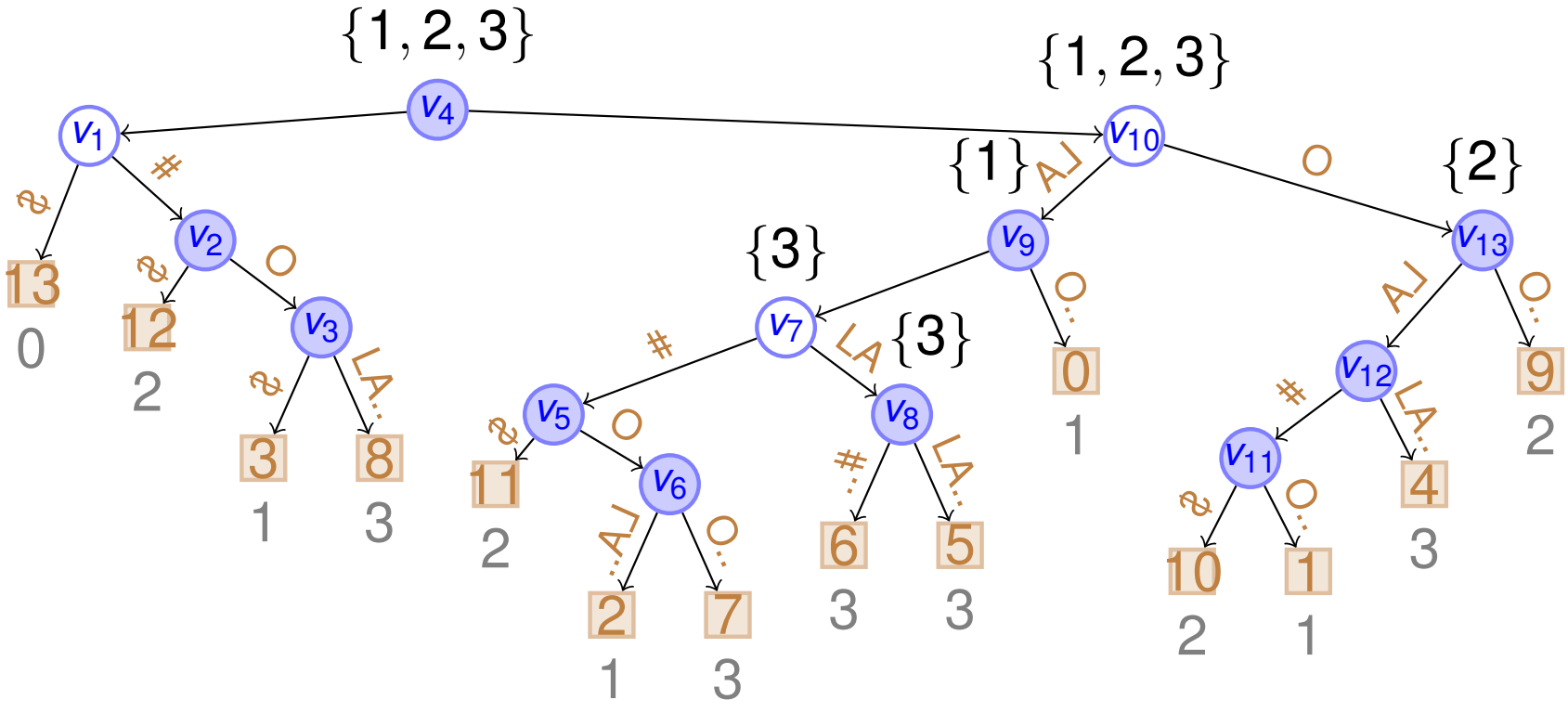
- For each 0 in H, record repeated doc ID ($O(n \log N)$ bits).
- Build WT over R.
- Map $[l, r]$ to *WTR* (via rank/select).
- Traverse *WTD* and *WTR* simultaneously.

Document Frequency for Subsets: Repetition Array



Space Reduction

- $\hat{R}$: omit entries in R, which belong to ST root

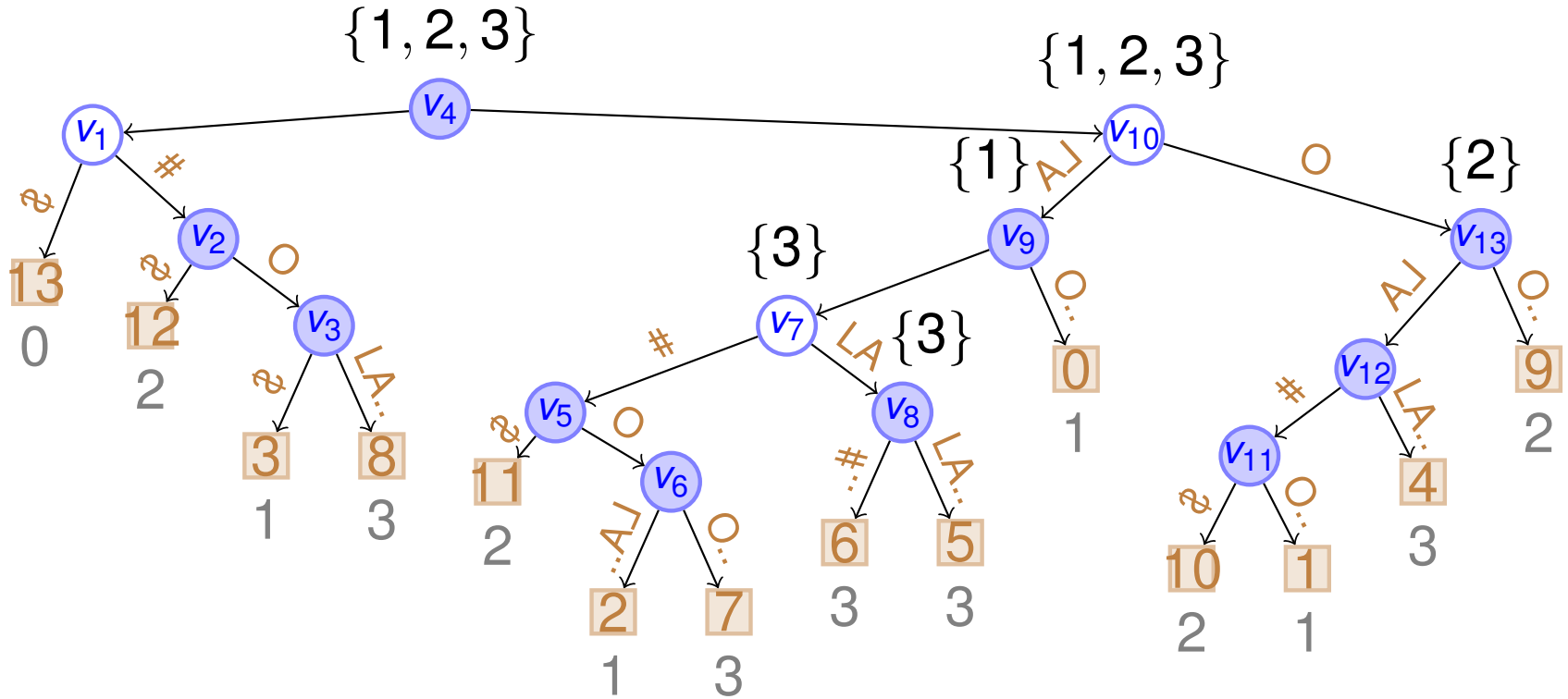


$H =$ 1 1 1 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1

$R =$ 3 3 1 2

Space Reduction

- $\hat{R}$: omit entries in R, which belong to ST root
- $\hat{R}^\ell$ and $\hat{D}^\ell$: If phrase length is restricted to ℓ, sorting intervals with common prefix of length $\geq \ell$ does not change correctness of method.



Space Reduction

- $\hat{R}$: omit entries in R, which belong to ST root
- $\hat{R}^\ell$ and $\hat{D}^\ell$: If phrase length is restricted to ℓ, sorting intervals with common prefix of length $\geq \ell$ does not change correctness of method.
- Both, WT-$\hat{R}^\ell$ and WT-D$^\ell$ contain frequency information. Given WT-$\hat{R}^\ell$,
  - omit duplicates for intervals with common prefix of length $\geq \ell$ in WT-D$^\ell$
  - sort intervals
  - in our example: $D^1 = \{0, 1, 2, 3, 1, 2, 3, 1, 2, 3\}$ instead of $D = \{0, 2, 1, 3, 2, 1, 3, 3, 3, 1, 2, 1, 3, 2\}$

Implementation

Based on SDSL components. Available at

`https://github.com/simongog/surf/`.

Includes engineered II implementation (block-max WAND + document reordering for better compression).

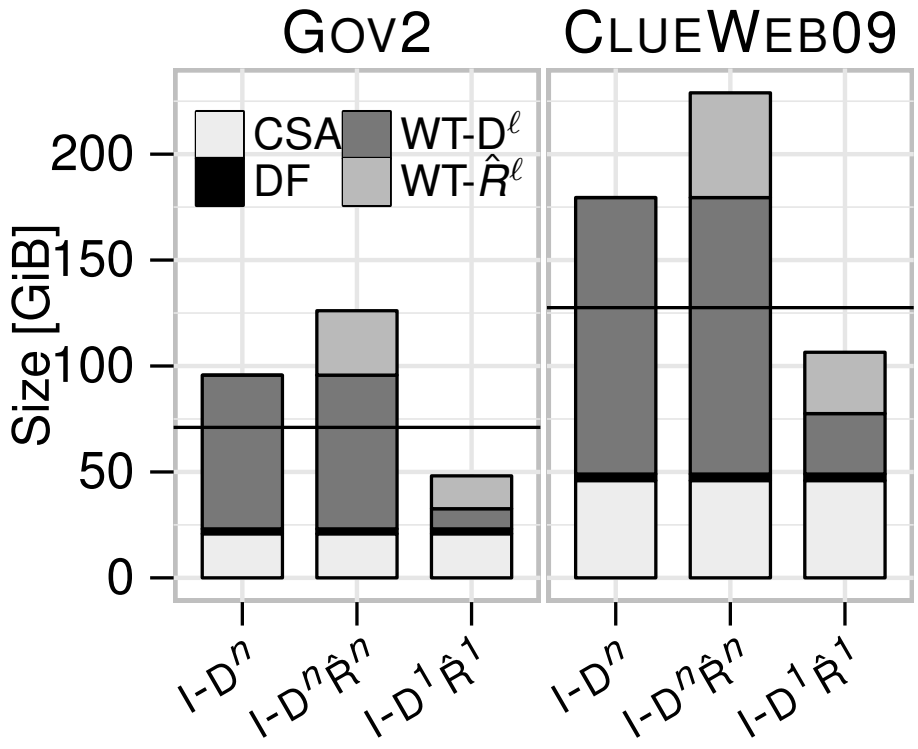
Collection Statistics

	GOV2	CLUEWEB09
n	23,468,782,575	40,579,891,952
N	25,205,179	50,220,423
n_{avg}	931	808
n_{min}	2	3
n_{max}	127,149	217,442
σ	39,177,922	90,411,635
$\lvert\mathcal{C}^{raw}\rvert$	$\approx$ 426 GiB	$\approx$ 1.2 TiB
$\lvert\mathcal{C}^{word}\rvert$	72 GiB	128 GiB

Notice

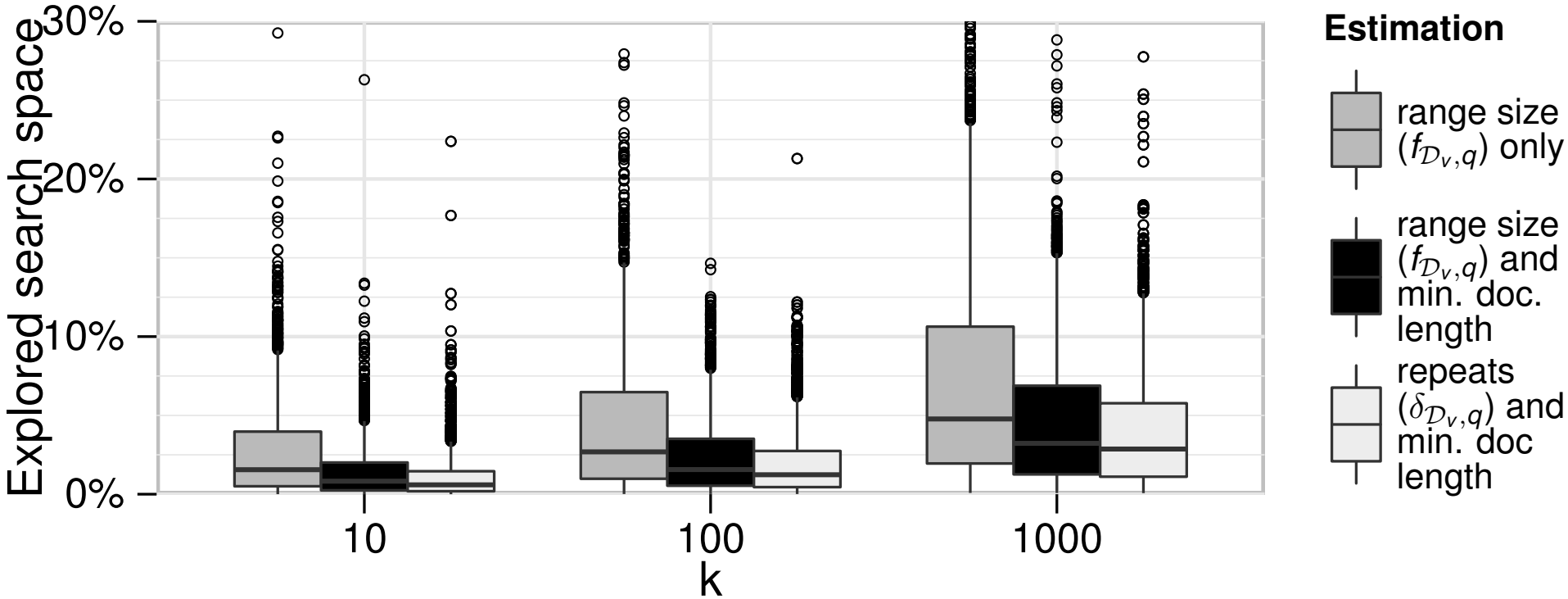
- Input is the word parsing $\mathcal{C}^{word}$ (generated by Indri)
- Queries from the TREC 2005/2006 efficiency track

Index Sizes

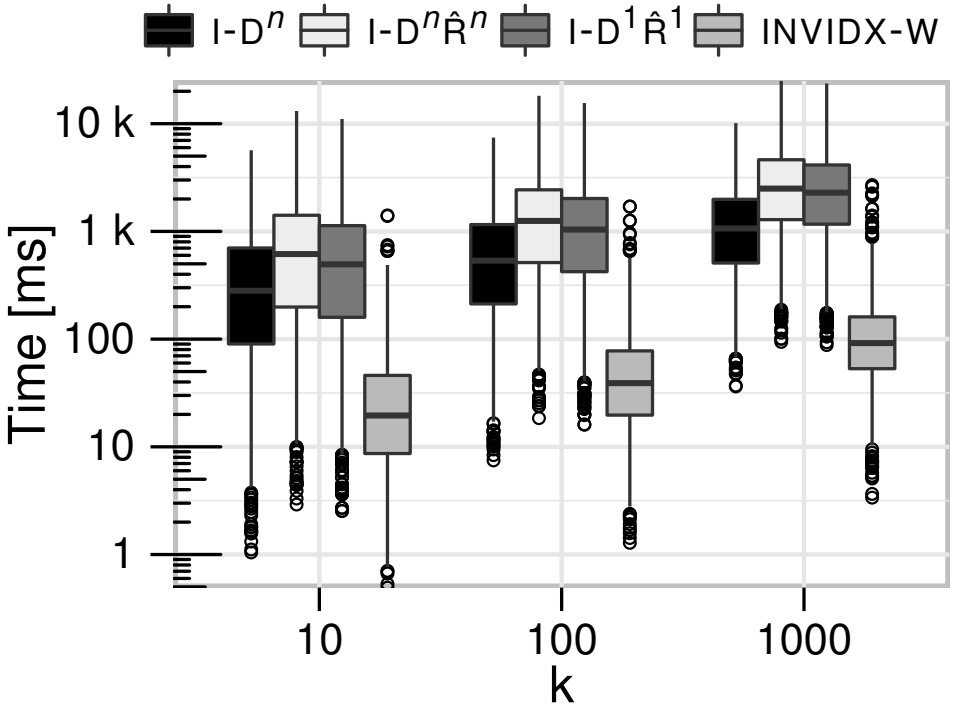


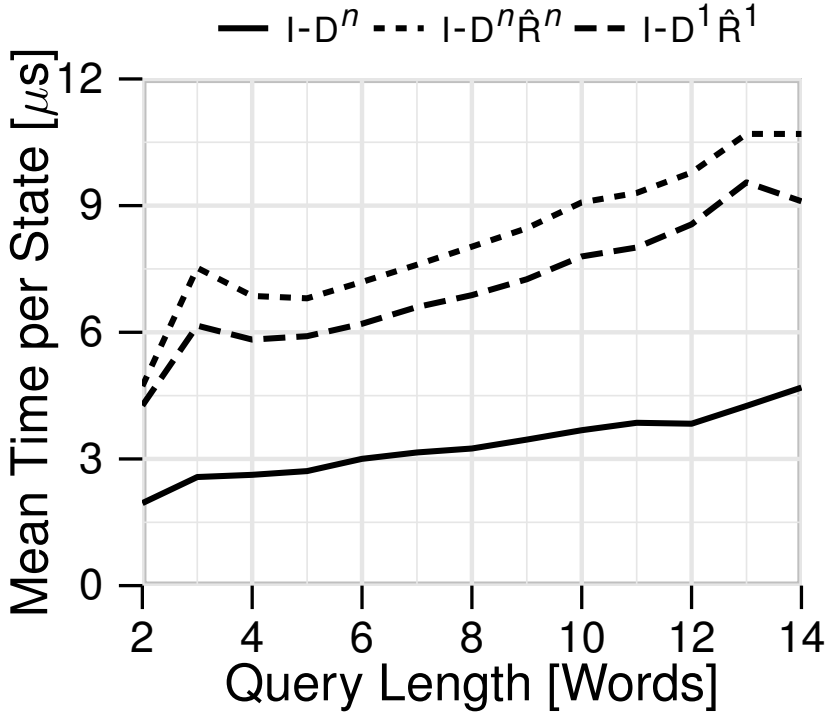
- Details: `http://go.unimelb.edu.au/6a4n`.
- Horizontal line: size of word parsing
- PII: typically 50%-60% (+original text), our II: 7 GiB for GOV2

Evaluated States



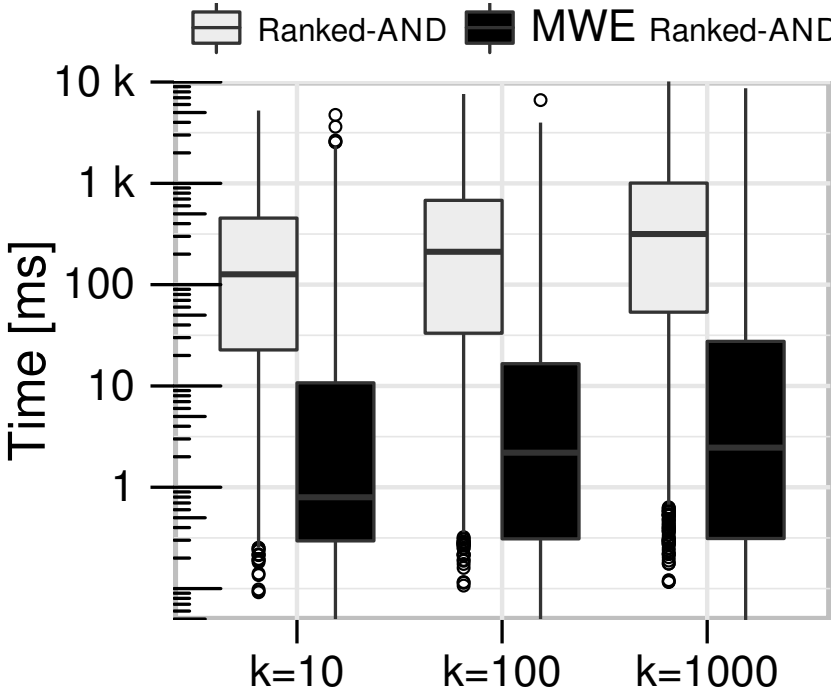
Query Times (1)

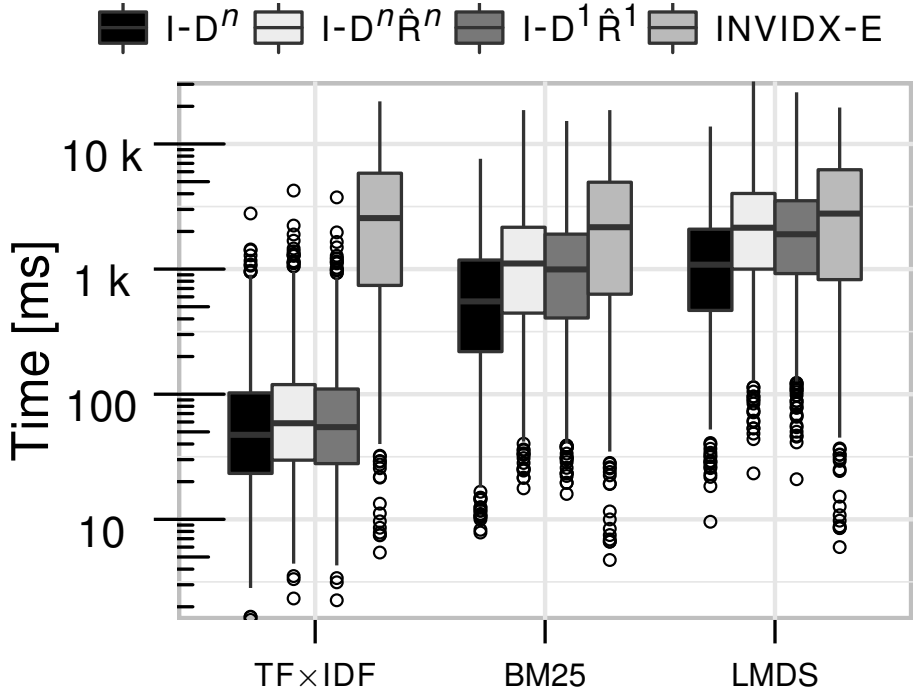




Left: BM25 Ranked-OR retrieval on GOV2. Right: Time per state.

Query Times (2)





Ranked-AND BM25 runtime for unparsed and MWE-parsed queries (left) and Ranked-OR runtime for different similarity measures and indexes (right).

Conclusion

- Extended GREEDY approach to multi-term queries and complex scoring functions
- Conceptional very simple search engine
- Flexible: ranked-and/or, scoring function
- II still better for bag of word queries (for precomputed scores, i.e. fixed scoring function)
- But: self-index based solution provides more functionality
  - phrase search (experiment with multi-word expressions)
  - text extraction
  - query completion (without query log)
- First self-index based system on scale
- Future work: WTs of higher arity, faster state processing


J. Shane Culpepper, Gonzalo Navarro, Simon J. Puglisi, and Andrew Turpin.
Top-*k* ranked document search in general text databases.
In *Proc. ESA*, pages 194–205, 2010.


Kunihiko Sadakane.
Succinct data structures for flexible text retrieval systems.
J. Discrete Alg., 5(1):12–22, 2007.