

Selectivity Estimation of Substring Queries



Kyuseok Shim
Seoul National University
<http://ee.snu.ac.kr/~shim>

1

Motivation



- `SELECT title`
`FROM papers`
`WHERE title LIKE '%string%' and`
`numOfpages < 13`
- Given several keywords, estimate number of documents having all keywords

2



Substring Selectivity Estimation Methods

- Markov-chain Approach
 - KVI Algorithm [krishnan, Vitter, Iyer: SIGMOD'96]
 - Complete conditional independence (CCI)
 - MO (Maximal Overlap) Algorithm [Jagadish, Ng, Srivastava:PODS'99]
 - Conditional dependence of a_j on the immediately preceding (maximal overlap) substring
 - Markov Estimator (ME)
 - QG Estimator
 - Selectivity of a s can never exceed that of s' for any substring s' of s
 - CRT Algorithm [Chaudhuri, Ganti, Gravano: ICDE'04]
 - Short Identifying Substring (SIS)
- Monte Carlo Approach
 - Min Hashing Algorithm [Chen, Korn, Koudas, Muthkrishnan: PODS'00]]



Markov-chain Approach

KVI Algorithm & MO Algorithm



Substring Selectivity Estimation

- Problem Definition
 - Given
 - A pruned count-suffix tree T with a prune threshold p
 - A substring query σ
 - Estimate the fraction C_σ/N
 - C_σ : count of substring query σ in the database
 - N : count associated with the root of T

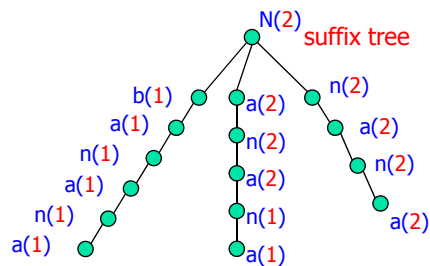


The Suffix Trees

- Suffix Tree
 - Stores not only strings but also all suffixes of each string
- Count-Suffix Tree
 - Does not store pointers to occurrence of the substrings
 - Just keeps a count C_α at the node corresponding to α
- Count in each node
 - The number of strings in the D containing α as a substring

An Example of a Suffix Tree

- 'banana', 'nana'
 - suffix tree $\Rightarrow C_{ana} = 2$
 - N denotes number of strings in D
 - $C_{null} = N = 2$



Pruned Count-Suffix Tree

- Pruned count-suffix tree (PST: T)
 - Prune away every node that has a $C_v \leq p$, where p is the pruned threshold
- Completion of a PST
 - A count-suffix tree is a completion of a PST T if T can be obtained by pruning the count-suffix tree
 - Same PST can be generated by pruning different count suffix trees
 - $C(T)$: set of all completions of T

Maximal Overlap

- For string $\beta = \alpha_1\alpha_2$
 - α_1 is a prefix of β
 - $\beta - \alpha_1$ gives the suffix α_2

- Maximal Overlap
 - Given $\beta_1 = \alpha_1\alpha_2$ and $\beta_2 = \alpha_2\alpha_3$, α_2 is maximal
 - Maximal overlap between a suffix of β_1 and a prefix of β_2
 - $\beta_1 \cap \beta_2 = \alpha_2$
 - $\beta_1 - \beta_1 \cap \beta_2 = \alpha_1$
 - $\beta_2 - \beta_1 \cap \beta_2 = \alpha_3$

Selectivity Estimation Algorithms

- $\Pr(\sigma)$: Selectivity of substring query σ
 - case 1: σ is found in the PST \mathcal{T}
 - $\Pr(\sigma) = C_\sigma / N$
 - case 2: σ is not found in the PST \mathcal{T}
 - we must estimate $\Pr(\sigma)$
 - This is essence of our **substring selectivity estimation** problem
 - Let $\sigma = \alpha_1 \dots \alpha_w$
 - $\Pr(\sigma) = \Pr(\alpha_w | \alpha_1 \dots \alpha_{w-1}) * \Pr(\alpha_1 \dots \alpha_{w-1})$

$$= \Pr(\alpha_w | \alpha_1 \dots \alpha_{w-1}) * \Pr(\alpha_{w-1} | \alpha_1 \dots \alpha_{w-2}) * \Pr(\alpha_1 \dots \alpha_{w-2})$$


$$= \left(\prod_{j=2}^w \Pr(\alpha_j | \alpha_1 \dots \alpha_{j-1}) \right) * \Pr(\alpha_1)$$

$$\frac{\Pr(\alpha_1 \dots \alpha_w)}{\Pr(\alpha_1 \dots \alpha_{w-1}) * \Pr(\alpha_1 \dots \alpha_{w-1})} = \Pr(\alpha_1 \dots \alpha_w) = \Pr(\sigma)$$

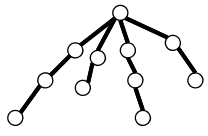
We cannot compute conditional probability because some suffixes are not in the PST
- ※ $\Pr(\alpha_j | \alpha_1 \dots \alpha_{j-1})$: the probability of occurrence of α_j given that the preceding string $\alpha_1 \dots \alpha_{j-1}$ has been observed.

KVI Algorithm

- Assume **complete conditional independence (CCI)**
 - $\Pr(\alpha_j | \alpha_1 \dots \alpha_{j-1}) \cong \Pr(\alpha_j)$
- Greedy parsing
 - Finds a sequence of strings $\alpha_1 \dots \alpha_w$ for some w such that
 - (i) $\sigma = \alpha_1 \dots \alpha_w$
 - (ii) α_1 is the longest prefix of σ that can be found in the PST \mathcal{T}
 - (iii) for all $j > 1$, α_j is the longest prefix of $(\sigma - \alpha_1 - \dots - \alpha_{j-1})$



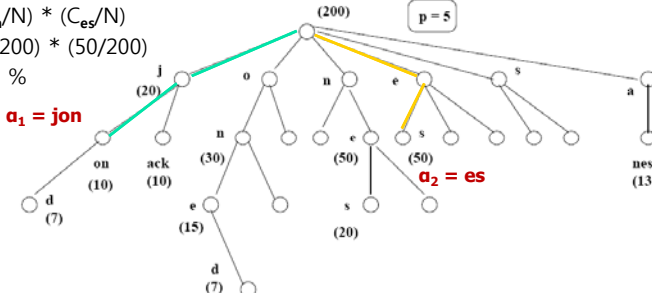
query string



PST

KVI Algorithm

- KVI Selectivity: $\Pr(\sigma) = \Pr(\alpha_1) * \dots * \Pr(\alpha_w)$
- Example [KVI Estimation], query $\sigma = \text{'jones'}$
 - $\Pr(\text{jones}) = \Pr(\text{'jon'}) * \Pr(\text{'es'|'jon'})$ (By greedy parsing)
 - $\cong \Pr(\text{'jon'}) * \Pr(\text{'es'})$ (By CCI)
 - $= (C_{\text{jon}}/N) * (C_{\text{es}}/N)$
 - $= (10/200) * (50/200)$
 - $= 1.25 \%$



MO (Maximal Overlap) Alg.

- Computes all maximal substring β_1, \dots, β_u of σ that can be found in the PST \mathcal{T}
 - β_1, \dots, β_u are selected such that

$$\sigma = \beta_1[\beta_2 - (\beta_1 \cap \beta_2)] \dots [\beta_u - (\beta_{u-1} \cap \beta_u)]$$
- Assume **conditional dependence of α_j on the immediately preceding (maximal overlap) substring**
 - $\Pr(\alpha_j | \alpha_1 \dots \alpha_{j-1}) \equiv \Pr(\alpha_j | \beta_{j-1} \cap \beta_j)$
 $= \Pr(\beta_j) / \Pr(\beta_{j-1} \cap \beta_j)$

$(\beta_{j-1} \cap \beta_j) \alpha_j = \beta_j$
 $\alpha_j = \beta_j - (\beta_{j-1} \cap \beta_j)$

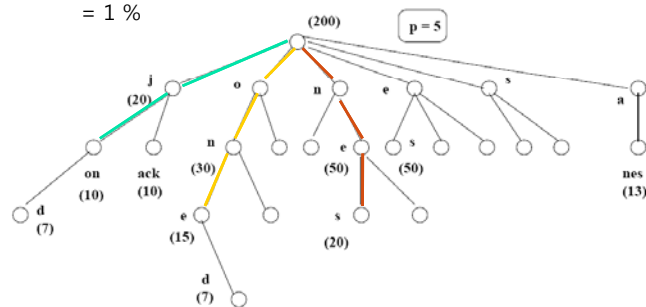
MO (Maximal Overlap) Alg.

- $\Pr(\sigma) = \Pr(\beta_1) * \Pr(\beta_2) / \Pr(\beta_1 \cap \beta_2) * \dots * \Pr(\beta_w) / \Pr(\beta_{w-1} \cap \beta_w)$
- Example [MO Parsing], query $\sigma = \text{'jones'}$
 - $\beta_1 = \text{'jon'}$, $\beta_2 = \text{'one'}$, $\beta_3 = \text{'nes'}$
 - $\beta_1 \cap \beta_2 = \text{'on'}$ and $\beta_2 \cap \beta_3 = \text{'ne'}$

MO (Maximal Overlap) Alg.

- Example [MO Estimation], query $\sigma = \text{'jones'}$
 - $\beta_1 = \text{'jon'}$, $\beta_2 = \text{'one'}$, $\beta_3 = \text{'nes'}$
 - $\Pr(\text{jones}) = \Pr(\text{'jon'}) * \Pr(\text{'e'|'jon'}) * \Pr(\text{'s'|'jone'})$
 $\equiv \Pr(\text{'jon'}) * \Pr(\text{'e'|'on'}) * \Pr(\text{'s'|'ne'})$
 $= (\Pr(\text{'jon'})) * (\Pr(\text{'one'})/\Pr(\text{'on'})) * (\Pr(\text{'nes'})/\Pr(\text{'ne'}))$
 $= (C_{\text{jon}}/N) * (C_{\text{one}}/C_{\text{on}}) * (C_{\text{nes}}/C_{\text{ne}})$
 $= 1 \%$

$$\begin{aligned} \Pr(\alpha_j | \alpha_1 \dots \alpha_{j-1}) \\ &\equiv \Pr(\alpha_j | \beta_{j-1} \circ \beta_j) \\ &= \Pr(\beta_j) / \Pr(\beta_{j-1} \circ \beta_j) \\ &= C_{\beta_j} / C_{\beta_{j-1} \circ \beta_j} \end{aligned}$$



Pseudo Code of KVI and MO Alg.s

Algorithm KVI

Input: a PST \mathcal{T} with prune threshold p , and root count N ; a substring query σ

Output: the estimate $KVI(\sigma)$

```
{ 1.  $i = 1$ ;
  2. While ( $\sigma$  not equal null) {
    2.1  $\gamma =$  the longest prefix of  $\sigma$  found in  $\mathcal{T}$ ;
    2.2 If ( $\gamma$  equal null) {
    2.3  $\alpha_i = \sigma[1]$ ;
    2.4  $\Pr(\alpha_i) = p/N$ ; }
    Else {
    2.5  $\alpha_i = \gamma$ ;
    2.6  $\Pr(\alpha_i) = C_{\alpha_i}/N$ ; }
    2.7  $\sigma = \sigma - \alpha_i$ ;
    2.8  $i = i + 1$ ; }
  3.  $KVI = \Pi_i \Pr(\alpha_i)$ ; return( $KVI$ );
}
```

Algorithm MO

Input: a PST \mathcal{T} with prune threshold p , and root count N ; a substring query σ

Output: the estimate $MO(\sigma)$

```
{ 1.  $i = 1$ ;  $\beta_0 =$  null;
  2. While ( $\sigma$  not equal null) {
    2.1  $\gamma =$  the longest prefix of  $\sigma$  found in  $\mathcal{T}$ ;
    2.2 If ( $\gamma$  equal null) {
    2.3  $\beta_i = \sigma[1]$ ;
    2.4  $\Pr(\beta_i) = p/N$ ;
    2.5  $i = i + 1$ ; }
    Else if ( $\gamma$  is not a substring of  $\beta_{i-1}$ ) {
    2.7  $\beta_i = \gamma$ ;
    2.8  $\Pr(\beta_i) = C_{\beta_i}/C_{\beta_{i-1} \circ \beta_i}$ ;
    2.9  $i = i + 1$ ; }
    2.10  $\sigma = \sigma - \sigma[1]$ ; }
  3.  $MO = \Pi_i \Pr(\beta_i)$ ; return( $MO$ );
}
```




MO vs. KVI

- KVI: Complete conditional independence assumption
- MO: partial conditional dependence assumption
 - It does not guarantee MO is always better than KVI
 - **Short memory property** - there exists a length L such that
 - If the length of preceding subsequences of the string $> L$, the conditional probability does not change substantially
- Theorem
 - Suppose the strings in \mathcal{D} exhibit the short memory property with length L
 - Let β_1, \dots, β_n be the maximal substrings on σ in \mathcal{T}
 - If $\forall i > 1 : \beta_{i-1} \cap \beta_i$ has length $\geq L$, then MO is a better estimate than KVI



MO vs. KVI

- Experimental Result
 - A real AT&T data about last names of over 100,000 employees
 - Positive Query - present in the un-pruned tree, but not in the PST \mathcal{T}
 - Use Relative Error
Error = (estimated count - actual count) / actual count
 - Negative Query - present neither in the un-pruned tree nor in the PST \mathcal{T}
 - Actual count would be 0
 - Use Standard error (the square root of mean squared error)

	Positive Queries (avg. relative error)	Negative Queries (avg. standard error)
MO	-28%	0.08
KVI	+326%	0.15



Markov-chain Approach

GE Estimator

19



Q-grams

- Extended Strings $\Rightarrow \text{ext}(s)$
 - prefixing s with '#' and suffixing it with '\$'
 - ex) $\text{ext}(\text{"seattle"}) = \text{"#seattle\$"}$
- Predicate Matching
 - a tuple t is said to satisfy or match a unit predicate "R.A like %s%" if s is a substring of $t[A]$
- Frequency
 - the number of tuples in R that match the unit predicate p
- Selectivity
 - $f(p) / |R|$



Q-grams

- Q-gram Table $\Rightarrow QT_q(R.A)$
 - q-gram: Any string of length q in $(\Sigma \cup \{\$, \#\})^*$
 - A lookup table with the frequency of each n -grams s_n where $1 \leq n \leq q$

- Q-gram Sequence $\Rightarrow Q_q(s)$
 - Ordered sequence of all (overlapping) q -grams
 - $Q_3(\text{'seattle' is})$ is $[\text{'sea'}$, 'eat' , 'att' , 'ttl' , 'tle']



QG Estimator

- Describes an upper bound on the selectivity of a unit predicate
- selectivity of a predicate $\%s\%$ can never exceed that of $\%s'\%$ for any substring s' of s
- QG estimator returns the minimum selectivity of a q -gram of string s
- Ex. Given below table, $QG(\%novel\%) = \min\{f(\text{nov}), f(\text{ove}), f(\text{vel})\} / 5$

<i>R.A</i>	Extended q -gram sequences ($q=3$)
novel	$[\#no, nov, ove, vel, el\$]$
article	$[\#ar, art, rti, tic, icl, cle, le\$]$
paper	$[\#pa, pap, ape, per, er\$]$
journal	$[\#jo, jou, our, urn, ma, nal, al\$]$
magazine	$[\#ma, mag, aga, gaz, azi, zin, ine, ne\$]$



Markov-chain Approach

CRT Algorithm

23



Short Identifying Substring Hypothesis

- **Identifying substring**
 - Consider a unit predicate R.A like %s%. We say that a substring s' of s is an (ϵ, β) identifying substring, for $0 \leq \epsilon < 1$ and $0 < \beta < 1$, if
 - (i) the selectivity of R.A like %s'% is no larger than $(1+\epsilon)$ times that of R.A like %s% and
 - (ii) $|s'| \leq \beta |s|$
- Ex. "ove" is a $(0, 0.6)$ identifying substring
 - Selectivity of %novel% coincides with that of %ove% ($\epsilon = 0$)
 - length of "ove" is $3 = 0.6 * 5$ ($\beta=0.6$)

Finding Minimal Identifying Substring

- Minimal identifying substring
 - Does not strictly contain another identifying substring of s
- We would like to correctly guess a "minimal"
- Once minimal identifying substring is found, we can use existing estimators (e.g. the Markov estimator)
 - Ex. $P(\text{'seattle'}) = P(\text{'eatt'}) = P(\text{'eat'}) * P(\text{'att'}|\text{'eat'})$
- However, if we only have limited statistics on frequencies of substrings
 - We cannot find the "minimal"
 - => multiple candidate identifying substrings

Short Identifying Substring Hypothesis

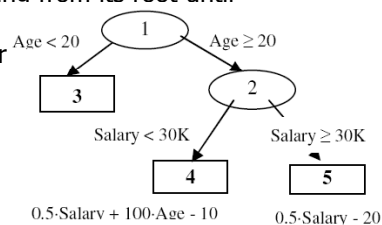
- Short Identifying Substring Hypothesis
 - A query string s usually has a "short" substring s' such that if an attribute value contains s' , then the attribute value almost always contain s as well
 - The selectivity of R.A like `%eatt%` is almost same as the selectivity of R.A like `%seattle%`
 - So, we use short identifying substrings to estimate the selectivity of query string instead of using full query string
- Selectivities of substrings are computed as follows:
 - Use Markov Estimator
 - Selectivities of length $m(>q)$ substrings can be obtained by multiplying conditional probabilities of q -grams in the q -gram table

Adjusting Minimal Identifying Substring

- When only limited statistics are available
 - We cannot guess the shortest (minimal) identifying substring
- So, we use another estimator
 - Guess several candidate identifying substrings for one of each possible length between q and $|s|$
 - Assign weights to each candidate
 - Combine their associated selectivities
- Weights are determined by a regression tree
 - Learn the characteristics of data and query workload

Regression Tree Overview

- Consider a relation R with numerical attributes X_1, \dots, X_m, Y
 - Attributes X_1, \dots, X_m are the predictor attributes
 - Y is a dependent attribute
 - $Y = f_n(X_1, \dots, X_m)$
- Given a tuple $[x_1, \dots, x_m, \text{NULL}]$
 - Traverse the regression tree RT starting from its root until reach a leaf node n
 - At n , $f_n(X_1, \dots, X_m)$ is the RT predictor of the Y value for the tuple
- Used for modeling dependencies between selectivities of a string and its substrings



Multiple Candidate Identifying Substrings

- Multiple Candidate Identifying Substrings
 - One for each value of substring length between q and $|s|$
 - For each length between q and $|s|$, find the substring of that length most likely to be an identifying substring

- Combination Function
 - Assigns weights to each selectivities of candidate identifying substrings
 - Weight
 - The weight of a substring of Length L depends on the probability that the length of the shortest identifying substring is equal to L

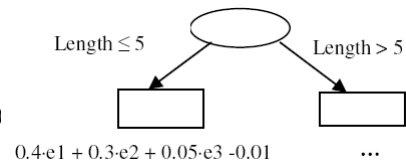
Choice of Candidate Identifying Substrings

- Choose one potential candidate identifying substring for each level
- Choice Rule
 - choose the substring at each level with the smallest (estimated) selectivity
 - The selectivity of a candidate identifying substring is guaranteed to be at least as high as that of the query string
 - Selectivities of candidate identifying substrings
 - at Level 0 : q-gram table
 - at Level 1~ : ME selectivity using q-gram table

Level 2	novel	W2	
Level 1	nove	ovel	W1
Level 0	nov	ove	vel W0

Regression-Tree Combination Function

- Assigns weights to selectivities of candidate identifying substrings at each level
 - learn the level weights from the data sets and expected query workload
- Selectivity of R.A like %s%
 - Weighted geometric mean
 - $x_0^{w_0} * \dots * x_n^{w_n} = \exp(w_0 \log x_0 + \dots + w_n \log x_n)$
 - Where x_m is selectivity of level m candidate identifying substring and w_n is weights at level m
- Ex. R.A like %novel%
 - $\exp(0.4 \log(\text{ME-Selectivity}(\%ove\%)) + 0.3 \log(\text{ME-Selectivity}(\%nove\%)) + 0.05 \log(\text{ME-Selectivity}(\%novel\%)) - 0.01)$



Monte Carlo Approach

Min Hash Algorithm

Min Hashing Method

Document 1. Peanut butter lover's club ...

Document 2. ... peanut stock...

Document 3. ...butter, the natural choice...

...

Document 100. ...

How many documents contain substring **peanut** but **not butter**?

- Boolean queries on substring predicates
 - Information Retrieval: Bibliographic search
 - Web searching: 40 millions per day at AltaVista
 - RDB queries

33

Motivation

- Store all correlations
 - Exponential (2^m) space to store correlation between substring predicates (m as number of substrings)
- Not to store correlation?
 - No! Correlation is important!
- The pitfalls in independence assumption

Document 1. **Peanut butter** lover's club ...

Document 2. ... **peanut** stock...

Document 3. ...**butter**, the natural choice...

...

Document 100. ...

25 times smaller than true count!

Independence Assumption:

$P(\text{peanut} \wedge \text{butter})$

$= P(\text{peanut}) * P(\text{butter})$

$= 2/100 * 2/100 = 0.04\%$

Derivation from Definition

- Definition of Min Hash

- $\Pr(\min \pi(A) = \pi(x) \mid x \in A) = \frac{1}{|A|}$

- for randomly chosen π

- For similarity calculation

- $\Pr(\min \pi(A \cup B) = \pi(x) \mid x \in A \cup B) = \frac{1}{|A \cup B|}$

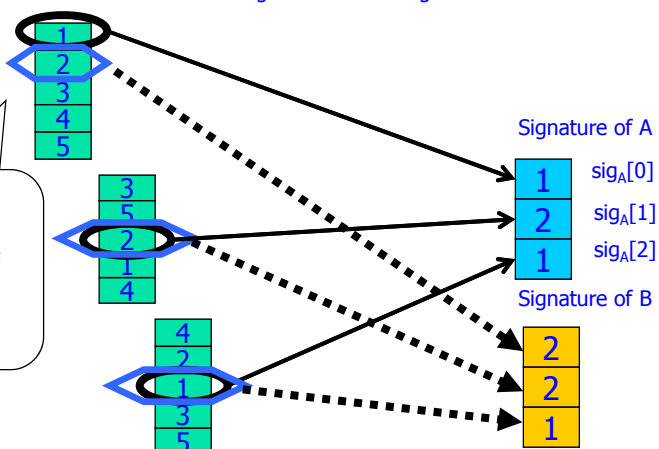
35

Signature Generation

Universe = {1,2,3,4,5}
A={1,2,3}, B={2,3,5}

Generate signatures with length 3

Randomly permute
universe 3 times



Computation of $|A \cap B|$

A's signature S_A

1	2	1
---	---	---

B's signature S_B

2	2	1
---	---	---

Definition r: # of pair-wise matches of S_A and S_B / length of signature

$$r = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{2}{3}$$

$$|A \cap B| = (r / (1 + r)) * (|A| + |B|)$$

$$= 2/3 / (1 + 2/3) * (3 + 3) = 2.4$$

Multiset Min Hashing

- Estimate $|A_1 \cap \dots \cap A_k|$
 - ρ_k : # of k-pair wise match / length of signature

$$\rho_k = \frac{|A_1 \cap \dots \cap A_k|}{|A_1 \cup \dots \cup A_k|}$$


- Estimate $|A_1 \cup \dots \cup A_k|$

$$\gamma = \frac{|A_j|}{|A_1 \cup \dots \cup A_k|}$$

$$\text{sig}(A_1 \cup \dots \cup A_k)[i] = \min(\text{sig}(A_1)[i], \dots, \text{sig}(A_k)[i])$$

- Estimate $|A_1 \cap \dots \cap A_k|$

$$|A_1 \cap \dots \cap A_k| = \rho_k |A_1 \cup \dots \cup A_k| = \rho_k \frac{|A_j|}{\gamma}$$




Quiz

	r1	r2	r3	r4	r5
π_1	1	5	3	2	4
π_2	3	2	1	4	5
π_3	4	3	5	1	2
π_4	5	4	2	3	1

- Let $A = \{r1, r3, r5\}$ and $B = \{r4, r5\}$.
- What is Min Hash Signatures for A and B?

39



Quiz

	r1	r2	r3	r4	r5
π_1	1	5	3	2	4
π_2	3	2	1	4	5
π_3	4	3	5	1	2
π_4	5	4	2	3	1

- Let $A = \{r1, r3, r5\}$ and $B = \{r4, r5\}$.
- What is Min Hash Signatures for A and B?
 - $\text{Sig}(A) = (1,1,2,1)$, $\text{sig}(B) = (2,4,1,1)$
- What is the estimated size of $A \cap B$ if I know $|A \cup B| = 5$?

40



Quiz

	r1	r2	r3	r4	r5
π_1	1	5	3	2	4
π_2	3	2	1	4	5
π_3	4	3	5	1	2
π_4	5	4	2	3	1

- Let $A = \{r_1, r_3, r_5\}$ and $B = \{r_4, r_5\}$.
- What is Min Hash Signatures for A and B?
 - $\text{Sig}(A) = (1, 1, 2, 1)$, $\text{sig}(B) = (2, 4, 1, 1)$
- What is the estimated size of $A \cap B$ if I know $|A \cup B| = 5$?
 - $|A \cap B| = \rho_k |A \cup B| = \frac{1}{4} * 5 = 5/4$
- What is the estimated size of $A \cup B$ if I know $|A| = 3$ and $|B| = 2$?

41



Quiz

	r1	r2	r3	r4	r5
π_1	1	5	3	2	4
π_2	3	2	1	4	5
π_3	4	3	5	1	2
π_4	5	4	2	3	1

- Let $A = \{r_1, r_3, r_5\}$ and $B = \{r_4, r_5\}$.
- What is Min Hash Signatures for A and B?
 - $\text{Sig}(A) = (1, 1, 2, 1)$, $\text{sig}(B) = (2, 4, 1, 1)$
- What is the estimated size of $A \cap B$ if I know $|A \cup B| = 5$?
 - $|A \cap B| = \rho_k |A \cup B| = \frac{1}{4} * 5 = 5/4$
- What is the estimated size of $A \cup B$ if I know $|A| = 3$ and $|B| = 2$?
 - $|A \cup B| = |A| / r = 3 * (4/3) = 4$
 - $\text{Sig}(A \cup B) = (1, 1, 1, 1)$, $r = 3/4$

42