Applied Databases

Lecture 20
Recap II

Sebastian Maneth

University of Edinburgh - March 30th, 2017
Recap II

1. Schemas, Normal Forms, SQL
2. TFIDF-ranking, string matching (KMP, automata, Boyer-Moore)
2. Relational DBs

1) explain, using examples, what a functional dependency (fd) is, and what a fd-redundancy is.

2) explain BCNF and how it removes fd-redundancies.

3) are there any “harmful” side-effects when transforming a table to BCNF?
1) explain, using examples, what a **functional dependency (fd)** is, and what a **fd-redundancy** is.

Let $S$ and $T$ be **non-empty sets** of attributes (column names).

A **table $R$** has a **functional dependency from $S$ to $T$**, if $R$’s projection to $S$ union $T$ gives a function from $S$ to $T$.

Such a function implies that for every $S$-tuple, there is at most one $T$-tuple in $R$. 
1) explain, using examples, what a functional dependency (fd) is, and what a fd-redundancy is.

Let $S$ and $T$ be non-empty sets of attributes (column names).

A table $R$ has a functional dependency from $S$ to $T$, if $R$’s projection to $S$ union $T$ gives a function from $S$ to $T$.

Such a function implies that for every $S$-tuple, there is at most one $T$-tuple in $R$.

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Functional dependencies? ("closed world assumption")
1) explain, using examples, what a *functional dependency (fd)* is, and what a *fd-redundancy* is.

Let \( S \) and \( T \) be **non-empty sets** of attributes (column names).

A **table** \( R \) has a **functional dependency from** \( S \) **to** \( T \), if \( R \)'s projection to \( S \) union \( T \) gives a function from \( S \) to \( T \).

Such a function implies that for every \( S \)-tuple, there is at most one \( T \)-tuple in \( R \).

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
<th>Functional dependencies? (&quot;closed world assumption&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>( X \rightarrow X ) ( XA \rightarrow X )</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>( X \rightarrow A ) ( XA \rightarrow A )</td>
</tr>
<tr>
<td>( X \rightarrow XA ) ( XA \rightarrow XA )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A \rightarrow A )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A \rightarrow X )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A \rightarrowXA )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1) explain, using examples, what a **functional dependency (fd)** is, and what a **fd-redundancy** is.

Let \( S \) and \( T \) be **non-empty sets** of attributes (column names).

A **table** \( R \) has a **functional dependency** from \( S \) to \( T \), if \( R \)'s projection to \( S \) union \( T \) gives a function from \( S \) to \( T \).

Such a function implies that for every \( S \)-tuple, there is at most one \( T \)-tuple in \( R \).

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
<th>Functional dependencies?</th>
<th>(&quot;closed world assumption&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>( X \rightarrow X )</td>
<td>( X \rightarrow A )</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>( X \rightarrow A )</td>
<td>FD's with disjoint ( S ), ( T )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( X \rightarrowXA )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( A \rightarrow A )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( A \rightarrow X )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( A \rightarrowXA )</td>
<td></td>
</tr>
</tbody>
</table>
1) explain, using examples, what a functional dependency (fd) is, and what a fd-redundancy is.

Let $S$ and $T$ be non-empty sets of attributes (column names). Functional dependency from $S$ to $T$: for every $S$-tuple, there is at most one $T$-tuple in $R$.

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

← how many functional dependencies?
1) explain, using examples, what a functional dependency (fd) is, and what a fd-redundancy is.

Let S and T be non-empty sets of attributes (column names).
Functional dependency from S to T:
for every S-tuple, there is at most one T-tuple in R.

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

← how many functional dependencies?

at most:
→ \((2^3 - 1) \times (2^3 - 1) = 7 \times 7 = 49\)

Which ones are excluded?

\[
\begin{align*}
A & \rightarrow Z, \ A \rightarrow XZ, \ A \rightarrow XA, \ A \rightarrow XAZ \\
X & \rightarrow Z, \ X \rightarrow AZ, \ X \rightarrow XA, \ X \rightarrow XAZ \\
XA & \rightarrow Z, \ XA \rightarrow AZ, \ XA \rightarrow xZ, \ XA \rightarrow XAZ
\end{align*}
\]
1) explain, using examples, what a functional dependency (fd) is, and what a fd-redundancy is.

Let S and T be non-empty sets of attributes (column names).
Functional dependency from S to T: for every S-tuple, there is at most one T-tuple in R.

A table R has fd-redundancy w.r.t. \( S \rightarrow T \), if R contains two distinct tuples with equal (S,T)-values.
1) explain, using examples, what a **functional dependency (fd)** is, and what a **fd-redundancy** is.

Let $S$ and $T$ be non-empty sets of attributes (column names).

**Functional dependency from $S$ to $T$:**
for every $S$-tuple, there is at most one $T$-tuple in $R$.

A table $R$ has **fd-redundancy w.r.t. $S \rightarrow T$**, if $R$ contains two distinct tuples with equal $(S,T)$-values.

<table>
<thead>
<tr>
<th>$X$</th>
<th>$A$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

← are there fd-redundancies?
1) explain, using examples, what a **functional dependency (fd)** is, and what a **fd-redundancy** is.

Let $S$ and $T$ be **non-empty sets** of attributes (column names).  
**Functional dependency from $S$ to $T$:**
for every $S$-tuple, there is at most one $T$-tuple in $R$.

A table $R$ has **fd-redundancy w.r.t. $S \rightarrow T$**, if $R$ contains **two distinct tuples** with equal $(S,T)$-values.

<table>
<thead>
<tr>
<th>$X$</th>
<th>$A$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

← are there fd-redundancies?

Yes:  
1) fd-redundancy wrt $X \rightarrow A$
2) fd-redundancy wrt $A \rightarrow X$
2) explain **BCNF** and how it removes **fd-redundancies**.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))
2) explain BCNF and how it removes fd-redundancies.

BCNF = if $S \rightarrow T$ is a functional dependency of $R$, then $S$ is a superkey.

(assuming $S$ disjoint $T$)

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

← in BCNF?
2) explain BCNF and how it removes *fd-redundancies*.

BCNF = if $S \rightarrow T$ is a functional dependency of $R$, then $S$ is a superkey.

(assuming $S$ disjoint $T$)

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

← in BCNF?

Yes: $X$ is superkey, and

$X \rightarrow A$ is the only functional dependency.
2) explain **BCNF** and how it removes **fd-redundancies**.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
| 1 | 2 | 6 | ← in BCNF?
2) explain BCNF and how it removes fd-redundancies.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

← in BCNF?

No: \( X \rightarrow A \) is fd, but \( X \) is not a superkey

\( A \rightarrow X \) is fd, but \( A \) is not a superkey
2) explain **BCNF** and how it removes **fd-redundancies**.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
2) explain **BCNF** and how it removes **fd-redundancies**.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))

In BCNF, there can be **no** fd-redundancies.

Why?
2) explain BCNF and how it removes fd-redundancies.

**BCNF** = if \( S \rightarrow T \) is a functional dependency of \( R \), then \( S \) is a superkey.

(assuming \( S \) disjoint \( T \))

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( X )</th>
<th>( A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

In BCNF, there can be no fd-redundancies.

Why?

Would imply that a tuple exists *twice* in \( R \) with same superkey-values.

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
3) are there any “harmful” side-effects when transforming a table to BCNF?

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
3) are there any “harmful” side-effects when transforming a table to **BCNF**?

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

![Transformation Diagram]
3) are there any “harmful” side-effects when transforming a table to BCNF?

Dependency $XZ \rightarrow A$ is lost
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;
List the result tuples for each of these SQL queries:

1) `SELECT * FROM R where b>a;`
2) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;`
3) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;`
4) `SELECT a FROM R UNION SELECT a FROM R;`
5) `SELECT * FROM R r1, R r2 where r1.b=r2.a;`
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

1) (1,2,5) and (1,3,5)
List the result tuples for each of these SQL queries:

1) `SELECT * FROM R where b>a;`
2) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;`
3) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;`
4) `SELECT a FROM R UNION SELECT a FROM R;`
5) `SELECT * FROM R r1, R r2 where r1.b=r2.a;`

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

1) (1, 2, 5) and (1, 3, 5)
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

1) (1,2,5) and (1,3,5)
2) (7)
List the result tuples for each of these SQL queries:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

1) (1, 2, 5) and (1, 3, 5)
2) (7)
3)
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

1) (1,2,5) and (1,3,5)
2) (7)
3) (6)
List the result tuples for each of these SQL queries:

1) \( \text{SELECT * FROM R where b > a;} \)
2) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b > r2.a;} \)
3) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b > r1.a;} \)
4) \( \text{SELECT a FROM R UNION SELECT a FROM R;} \)
5) \( \text{SELECT * FROM R r1, R r2 where r1.b = r2.a;} \)

1) \((1, 2, 5)\) and \((1, 3, 5)\)
2) \((7)\)
3) \((6)\)
4)
List the result tuples for each of these SQL queries:

1) \( \text{SELECT * FROM R where b>a;} \)

2) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;} \)

3) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;} \)

4) \( \text{SELECT a FROM R UNION SELECT a FROM R;} \)

5) \( \text{SELECT * FROM R r1, R r2 where r1.b=r2.a;} \)

1) \((1, 2, 5) \text{ and } (1, 3, 5)\)

2) \((7)\)

3) \((6)\)

4) \((1) \text{ and } (2)\)
**SQL**

List the result tuples for each of these *SQL* queries:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

1) `SELECT * FROM R where b>a;`
2) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;`
3) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;`
4) `SELECT a FROM R UNION SELECT a FROM R;`
5) `SELECT * FROM R r1, R r2 where r1.b=r2.a;`

1) `(1, 2, 5)` and `(1, 3, 5)`
2) `(7)`
3) `(6)`
4) `(1)` and `(2)`

5)
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

1) \((1, 2, 5)\) and \((1, 3, 5)\)

2) \((7)\)

3) \((6)\)

4) \((1)\) and \((2)\)

5) \((1, 2, 5, 2, 2, 6)\) and \((2, 2, 6, 2, 2, 6)\)
List the result tuples for each of these SQL queries:

1) `SELECT * FROM R where b>a;`
2) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;`
3) `SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;`
4) `SELECT a FROM R UNION SELECT a FROM R;`
5) `SELECT * FROM R r1, R r2 where r1.b=r2.a;`

Give SQL queries for

(a) all values (with duplicates) in the entire table R
(b) all distinct values in the entire table R, with their frequencies
(c) all distinct b-values in R, that are smaller than the average over all values (with duplicates) in the entire R.
List the result tuples for each of these SQL queries:

1) `SELECT * FROM R where b > a;`
2) `SELECT COUNT(*) FROM R r1, R r2 where r1.b > r2.a;`
3) `SELECT COUNT(*) FROM R r1, R r2 where r1.b > r1.a;`
4) `SELECT a FROM R UNION SELECT a FROM R;`
5) `SELECT * FROM R r1, R r2 where r1.b = r2.a;`

Give SQL query for

(a) all values (with duplicates) in the entire table R
List the result tuples for each of these SQL queries:

1) \( \text{SELECT} \) * FROM R \( \text{where} \) b>a;
2) \( \text{SELECT} \) COUNT(*) FROM R r1, R r2 \( \text{where} \) r1.b>r2.a;
3) \( \text{SELECT} \) COUNT(*) FROM R r1, R r2 \( \text{where} \) r1.b>r1.a;
4) \( \text{SELECT} \) a FROM R UNION \( \text{SELECT} \) a FROM R;
5) \( \text{SELECT} \) * FROM R r1, R r2 \( \text{where} \) r1.b=r2.a;

Give SQL query for

(a) all values (with duplicates) in the entire table R

\[
\rightarrow \text{SELECT a FROM R UNION ALL} \\
\text{SELECT b FROM R UNION ALL} \\
\text{SELECT c FROM R;}
\]
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

Give SQL query for

(b) all distinct values in the entire table R, with their frequencies
List the result tuples for each of these SQL queries:

1) SELECT * FROM R where b>a;
2) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3) SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4) SELECT a FROM R UNION SELECT a FROM R;
5) SELECT * FROM R r1, R r2 where r1.b=r2.a;

Give SQL query for

(b) all distinct values in the entire table R, with their frequencies

→ SELECT a,COUNT(a) FROM 
  (SELECT a FROM R UNION ALL
   SELECT b FROM R UNION ALL
   SELECT c FROM R) z
  GROUP BY a;
List the result tuples for each of these SQL queries:

1. SELECT * FROM R where b>a;
2. SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;
3. SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;
4. SELECT a FROM R UNION SELECT a FROM R;
5. SELECT * FROM R r1, R r2 where r1.b=r2.a;

Give SQL query for

(c) all distinct b-values in R, that are smaller than the average over all values (with duplicates) in the entire R.
List the result tuples for each of these SQL queries:

1) \( \text{SELECT * FROM R where b>a;} \)
2) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b>r2.a;} \)
3) \( \text{SELECT COUNT(*) FROM R r1, R r2 where r1.b>r1.a;} \)
4) \( \text{SELECT a FROM R UNION SELECT a FROM R;} \)
5) \( \text{SELECT * FROM R r1, R r2 where r1.b=r2.a;} \)

Give SQL query for (c) all distinct b-values in R, that are smaller than the average over all values (with duplicates) in the entire R.

\[ \text{SELECT DISTINCT b FROM R WHERE b< (SELECT AVG(a) FROM (SELECT a FROM R UNION ALL SELECT b FROM R UNION ALL SELECT c FROM R) z);} \]
TFIDF Ranking
TFIDF Ranking

Assume casefolding and stemming. We only care about these words:

   big, house, keep, night, old

1) make a table of term frequencies of these words (rows=words, columns=docs)
2) normalize by dividing column-wise by maximum
3) compute IDF of each word w as $\log_{10}(N/df_w)$
4) multiply normalized term frequencies by IDF, to obtain TFIDF table.
5) compute cosine similarity between doc-2 and “big old house”
Assume casefolding and stemming. We only care about these words:

   big, house, keep, night, old

1) make a table of term frequencies of these words (rows=words, columns=docs)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>house</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keep</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>night</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>big</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>house</td>
<td>1/2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keep</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>night</td>
<td>1/3</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>1/3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) normalize by dividing column-wise by maximum
3) compute IDF of each word w as \( \log_{10}(N/df_w) \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \log(6/2)=.477 )</td>
</tr>
<tr>
<td>house</td>
<td>1/2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(.477 )</td>
</tr>
<tr>
<td>keep</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \log(6/5)=.079 )</td>
</tr>
<tr>
<td>night</td>
<td>1/3</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td></td>
<td></td>
<td>( \log(6/3)=.301 )</td>
</tr>
<tr>
<td>old</td>
<td>1/3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>( \log(6/4)=.176 )</td>
</tr>
</tbody>
</table>
4) multiply normalized term frequencies by IDF, to obtain TFIDF table.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td></td>
<td>.477</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
</tr>
<tr>
<td>house</td>
<td>.239</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
</tr>
<tr>
<td>keep</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
</tr>
<tr>
<td>night</td>
<td>.100</td>
<td></td>
<td>.301</td>
<td>.201</td>
<td></td>
<td></td>
<td>.301</td>
</tr>
<tr>
<td>old</td>
<td>.059</td>
<td>.176</td>
<td>.176</td>
<td>.176</td>
<td></td>
<td></td>
<td>.176</td>
</tr>
</tbody>
</table>
5) compute \textbf{cosine similarity} between doc-2 and “big old house”

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>.477</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
</tr>
<tr>
<td>house</td>
<td>.239</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
</tr>
<tr>
<td>keep</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
</tr>
<tr>
<td>night</td>
<td>.100</td>
<td></td>
<td>.301</td>
<td>.201</td>
<td></td>
<td></td>
<td>.301</td>
</tr>
<tr>
<td>old</td>
<td>.059</td>
<td>.176</td>
<td>.176</td>
<td>.176</td>
<td></td>
<td></td>
<td>.176</td>
</tr>
</tbody>
</table>
5) compute **cosine similarity** between doc-2 and “big old house”

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>.477</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
<td>.477</td>
</tr>
<tr>
<td>house</td>
<td>.239</td>
<td>.477</td>
<td></td>
<td></td>
<td></td>
<td>.477</td>
<td>.477</td>
</tr>
<tr>
<td>keep</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td>.079</td>
<td></td>
</tr>
<tr>
<td>night</td>
<td>.100</td>
<td></td>
<td>.301</td>
<td>.201</td>
<td></td>
<td>.301</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>.059</td>
<td>.176</td>
<td>.176</td>
<td>.176</td>
<td></td>
<td>.176</td>
<td>.176</td>
</tr>
</tbody>
</table>

\[
\text{cos-sim}(Q,d2) = (\ .477 \times .477 + .239 \times .477 + .176 \times .176) / \\
(\sqrt{.477^2 + .239^2 + .176^2}) \times \sqrt{.477^2 + .477^2 + .176^2}) \\
= .3725 / (0.5618 \times 0.6972) = 0.9510
\]
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching Automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching Automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

1) → size of matching automaton is $|P||S|$ which can be $|P|^2$ ($S =$ alphabet)

→ KMP table has only $|P|$-many entries.

→ automaton uses one look-up per text-symbol, i.e., $O(|T|)$ matching time

→ KMP may require several look-ups per text-symbol (at most $(\log |P|)$–many)
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching Automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

2)

[Diagram of the Matching Automaton for abaaba]

aababa
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching Automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

2)

→ blue edges w.o. label means “else” = “any other letter”
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

3) KMP table = longest prefix that is proper suffix (up to current character) and such that the next letter (if exists) is different (−1” if such a prefix not exist)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
</table>
1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

3) KMP table = longest prefix that is proper suffix (up to current character) and such that the next letter (if exists) is different ("–1" if such a prefix not exist)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

3) KMP table = longest prefix that is proper suffix (up to current character) and such that the next letter (if exists) is different (“–1” if such a prefix not exist)

```
    | a | b | a | a | b | a |
---|---|---|---|---|---|---|
 0  | 0 | -1|   |   |   |   |
```
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

3) KMP table = longest prefix that is proper suffix (up to current character) and such that the next letter (if exists) is different ("–1" if such a prefix not exist)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

3) KMP table = longest prefix that is proper suffix (up to current character) and such that the next letter (if exists) is different (−1 if such a prefix not exist)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>3</td>
</tr>
</tbody>
</table>
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

4) If mismatch with $P[m]$ aligned to $z$ in $T$, shift RIGHT by $R(z)$.

$$R(z) = \text{distance from the right-most (non-last) “z” in } P \text{ to the end of } P$$

(and $|P|$ if there is no occurrence)
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

4) If mismatch with \( P[m] \) aligned to \( z \) in \( T \), shift RIGHT by \( R(z) \).

\[ R(z) = \text{distance from the right-most (non-last) “z” in } P \text{ to the end of } P \]

(and \( |P| \) if there is no occurrence)

#comparisons = 4
String Matching

1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string ababaaba?

4) If mismatch with \( P[m] \) aligned to \( z \) in \( T \), shift RIGHT by \( R(z) \).

\[
R(z) = \text{distance from the right-most (non-last) “z” in } P \text{ to the end of } P
\]
\( \text{ (and } |P| \text{ if there is no occurrence) } \)

\[
\begin{array}{cccccccc}
\text{a} & \text{a} & \text{b} & \text{a} & \text{b} & \text{a} & \text{a} & \text{b} \\
\text{a} & \text{b} & \text{a} & \text{a} & \text{b} & \text{a} \\
\text{a} & \text{b} & \text{a} & \text{a} & \text{b} & \text{a} \\
\text{a} & \text{b} & \text{a} & \text{a} & \text{b} & \text{a} \\
\end{array}
\]

\( \rightarrow R(a) = 2 \)
\( \rightarrow \) shift RIGHT by 2

\#	ext{comparisons} = 4 + 1
1) explain the difference between the Matching Automaton and KMP.
2) draw the Matching automaton for the string abaaba
3) give the KMP table for abaaba
4) how many comparisons does Horspool need for this pattern on the string aababaaba?

4) If mismatch with \( P[m] \) aligned to \( z \) in \( T \), shift RIGHT by \( R(z) \).

\[ R(z) = \text{distance from the right-most (non-last) "z" in } P \text{ to the end of } P \]

(and \( |P| \) if there is no occurrence)

\[
\begin{array}{cccccccccc}
  a & a & b & a & b & a & a & b & a & a \\
  a & b & a & a & b & a & a & b & a & a \\
  a & b & a & a & b & a & a & b & a & a \\
  a & a & b & a & b & a & a & b & a & a \\
\end{array}
\]

Mismatch with “b” aligned to \( P[m] \).
\[ \rightarrow \text{ shift by 1} = R(b) \]

\#comparisons = 4 + 1 + 6 = 11
END
Lecture 20

All the best with the exam!!
Remember: no lectures next week!