DAR Assignment 1

Automated Ranking of Queries

Utrecht University

April 30, 2025

Table of Contents

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
    - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. “Automated Ranking of Database Query Results”. In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.
- Write two programs in C#/Python (work in groups of two):

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.
- Write two programs in C#/Python (work in groups of two):

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.
- Write two programs in C#/Python (work in groups of two):
  - **Preprocessing/metadatabase:** used to preprocess the data/workload. A metadatabase is created and filled in.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
    - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.
- Write two programs in C#/Python (work in groups of two):
    - **Preprocessing/metadatabase:** used to preprocess the data/workload. A metadatabase is created and filled in.

Description Assignment 1

- **Implement article:** Sanjay Agrawal et al. "Automated Ranking of Database Query Results". In: *Microsoft Research/Stanford* (2003).
  - Read the article thoroughly!
- **Goal:** answering the many answers and the empty answers problem efficiently.
- **Reason:** A popular paradigm in Information Retrieval (IR) is to rank and return the most relevant results.
- Write two programs in C#/Python (work in groups of two):
  - **Preprocessing/metadatabase:** used to preprocess the data/workload. A metadatabase is created and filled in.
  - **Query/search program:** used to give the answer to the asked queries by the user, ideally with a top-k algorithm.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.
- **A C#/Python Search Program:** program using the metadabase program, which returns (a top-k of) the query asked by the user.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.
- **A C#/Python Search Program:** program using the metadabase program, which returns (a top-k of) the query asked by the user.

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.
- **A C#/Python Search Program:** program using the metadabase program, which returns (a top-k of) the query asked by the user.
- A quality **report** containing a class diagram of the search program (max. 6 pages).

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.
- **A C#/Python Search Program:** program using the metadabase program, which returns (a top-k of) the query asked by the user.
- A quality **report** containing a class diagram of the search program (max. 6 pages).

Delivarables (emphasized in bold)

- **A C#/Python Metadabase Program:** contains SQL statements for the creation of the metadatabase.
  Two text files are created:
  - **metadb.txt:** SQL CREATE TABLE statements for creating the metadatabase.
  - **metaload.txt:** SQL INSERT INTO statements for filling in the metadatabase.
- **A C#/Python Search Program:** program using the metadabase program, which returns (a top-k of) the query asked by the user.
- A quality **report** containing a class diagram of the search program (max. 6 pages).
- Give a **demo** of your program to one of the two TAs.

Some concepts

- Some concepts from IR can be used for DB (database). Others must be redefined.
  - For example, TF cannot be translated, but IDF can.
- One database table R with categorical and numerical attributes $\{A_1, A_2, \ldots, A_m\}$ and tuples $\{T_1, T_2, \ldots, T_n\}$.
- Conjunctive equality queries of the form

  SELECT * FROM R WHERE C_1 AND ... AND C_m,

  where each C_k is of the form $A_k = q_k$, with q_k the value of k.
- More advanced:

  A_k IN $\{q_{k1}, q_{k2}, ..., q_{kl}\}$

Example of organizing the metadatabase

- Numerical values
- Categorical values
- Think thoroughly about which attributes are categorical and which are numerical.
- The tables in the metadatabase will contain categorical and numerical attributes. Of course, there is more to the metadatabase!
- You are in charge of preprocessing the data/workload.

Useful concepts for metadatabase (see article for details)

- IDF similarities: a database ranking function containing categorical and numerical data. It is a measure of how rare a term is.
  - Formulas differ for categorical and numerical data (see the article for definitions).
  - Downside: smaller IDF assigned to attribute values with higher occurrence in the database. However, sometimes the relevance of the data depends on other factors.

Useful concepts for metadatabase (see article for details)

- The similarity coefficient is $S_k(u, v) = \begin{cases} IDF_k(u) & \text{if u = v} \\ 0 & \text{else} \end{cases}$ with u the value of A_k in the query, and v the value of A_k in the tuple.
- For a tuple $T = < t_1, t_2, \ldots, t_n >$ and a query $Q = < q_1, q_2, \ldots, q_m >$, the similarity between T and Q is the sum of the similarity coefficients:

$$sim(T, Q) = \sum_{k=1}^{m} S_k(t_k, q_k).$$

Useful concepts for metadatabase (see article for details)

- **QF similarities:** collecting the workload (i.e., past usage patterns) on the database, which is useful for ranking.
  - The importance of attribute values is determined by frequency occurrence. How many queries are executed with
    `SELECT ... FROM ... WHERE` $A_k = q_k$ `AND ... ?`
  - The query frequency $QF_k(q_k)$ (i.e., the popularity measure of q_k) is defined as $\frac{RQF_k(q_k)}{RQFMax_k}$ where $RQF_k(q_k)$ is the raw frequency of the value q_k of the attribute A_k that occurs in the workload and $RQFMax_k$ is the raw frequency of the value that occurs the most frequently in the workload.
  - The similarity coefficient is $S_k(t_k, q_k) = \begin{cases} QF_k(q_k) & q_k = t_k \\ 0 & \text{else} \end{cases}$

Useful concepts for metadatabase (see article for details)

- Example QF(q): Consider a workload that contains the following values of the attribute 'type': $\{SUV, Sports, Minivans, SUV, SUV, Sports\}$. Then, QF(Sports) $= \frac{RQF(Sports)}{RQFMAX} = \frac{2}{3}$.
- More **sophisticated** QF similarities: **Jaccard coefficient**.
- Most of the focus so far has been on categorical data. Please also pay attention to **numerical data**!

Jaccard coefficient

- The Jaccard coefficient measures the similarity between sets $W(t)$ and $W(q)$ (IN-clause):

$$J(W(t), W(q)) = \left| \frac{W(t) \cap W(q)}{W(t) \cup W(q)} \right|$$

- $W(v)$ is the subset of the queries in the workload where value v takes place in the IN-clause for a specific attribute.
- If pairs of values often occur together, they are likely similar: `brand IN (Lamborghini, Ferrari)`
- The similarity coefficient between t and q (the k subscript is dropped) can now be defined as this Jaccard coefficient that is scaled, for instance, by the QF factor:

$$S(t, q) = J(W(t), W(q))QF(q).$$

Jaccard coefficient example

- $Q_1 = \{\text{Opel, Citroën, Ford}\}$
- $Q_2 = \{\text{Peugeot, Audi, Opel}\}$
- $Q_3 = \{\text{Ford, Renault, Citroën, Mazda}\}$

- $W(Opel) = ...?$
- $W(Audi) = ...?$
- $W(Ford) = ...?$

Jaccard coefficient example

- $Q_1 = \{\text{Opel, Citroën, Ford}\}$
- $Q_2 = \{\text{Peugeot, Audi, Opel}\}$
- $Q_3 = \{\text{Ford, Renault, Citroën, Mazda}\}$

- $W(Opel) = ...?$
- $W(Audi) = ...?$
- $W(Ford) = ...?$

Jaccard coefficient example

- $Q_1 = \{\text{Opel, Citroën, Ford}\}$
- $Q_2 = \{\text{Peugeot, Audi, Opel}\}$
- $Q_3 = \{\text{Ford, Renault, Citroën, Mazda}\}$

- $W(Opel) = \{Q_1, Q_2\}$
- $W(Audi) = \{Q_2\}$
- $W(Ford) = \{Q_1, Q_3\}$

- $J(W(Opel), W(Audi)) = ...?$
- $J(W(Opel), W(Ford)) = ...?$
- $J(W(Audi), W(Ford)) = ...?$

Jaccard coefficient example

- $Q_1 = \{\text{Opel, Citroën, Ford}\}$
- $Q_2 = \{\text{Peugeot, Audi, Opel}\}$
- $Q_3 = \{\text{Ford, Renault, Citroën, Mazda}\}$

- $W(Opel) = \{Q_1, Q_2\}$
- $W(Audi) = \{Q_2\}$
- $W(Ford) = \{Q_1, Q_3\}$

- $J(W(Opel), W(Audi)) = ...?$
- $J(W(Opel), W(Ford)) = ...?$
- $J(W(Audi), W(Ford)) = ...?$

Jaccard coefficient

- $Q_1 = \{\text{Opel, Citroën, Ford}\}$
- $Q_2 = \{\text{Peugeot, Audi, Opel}\}$
- $Q_3 = \{\text{Ford, Renault, Citroën, Mazda}\}$

- $W(Opel) = \{Q_1, Q_2\}$
- $W(Audi) = \{Q_2\}$
- $W(Ford) = \{Q_1, Q_3\}$

- $J(W(Opel), W(Audi)) = \frac{1}{2}$
- $J(W(Opel), W(Ford)) = \frac{1}{3}$
- $J(W(Audi), W(Ford)) = 0$

Search program

- The goal of the search program is to give the best results of the database based on the asked query by the user, preferably with a top-k algorithm.
    - Examples of asked queries:

      `k = 6, brand = ‘volkswagen’;`

      `cylinders = 4, brand = ‘ford’.`
- Problems while searching (keep similarities in mind):
    - Many answers problem: if the query is not specific enough.
    - Empty answers problem: if the query is too specific.
- Make sure to tackle these problems!

Practical recommendations

Evaluation:

1. Deal with similarity properties. Pay also attention to the numerical attributes. Demonstrate your findings (max 8/10).
2. Use sophisticated techniques to find value similarities (+1 max).
3. Use sophisticated techniques for top-k calculations (+1 max).

See the evaluation form for more details.

Practical recommendations

- The first practicum will cover a small demonstration of a template in C# or Python by the TAs. The templates serve as a guideline to get an idea for the design of the assignment. It is not necessary to fill it in. **We advise you to write your own programs.**
  - The demonstration of the template in C# will take place in BBG 2.14 and the one in Python in BBG 2.09.
- The next lecture is on top-k algorithms and frequent itemsets! Both could be beneficial for this assignment.