Similarity Search
Lab 1 – Introduction to Project Proposals

Nikolaus Augsten

Free University of Bozen-Bolzano
Faculty of Computer Science
DIS

Lab 1 – March 8, 2012
Proposal 1: \(q\)-Gram Filter for String Edit Distance

- Implement the edit distance for strings.
- Implement the \(q\)-gram distance for strings (database implementation).
- Perform an approximate join using the edit distance and use the \(q\)-gram distance as a filter.
- Experiments:
  - compare the efficiency of the edit and the \(q\)-gram distance (vary the string length and the number of strings);
  - runtime performance of the edit distance with and without filter;
  - effectiveness of the filter on real world datasets.

The final report defines the problem, shortly introduces edit distance and \(q\)-gram distance, explains the filter mechanism and presents the experimental results. The report includes a usage manual for the software-deliverable.
Implement the tree edit distance.
Implement the binary branch filter (a lower bound of the tree edit distance).
Experiments:
- runtime performance of the tree edit distance with and without filter;
- effectiveness of the filter on example datasets.

The final report defines the problem, shortly introduces the tree edit distance and the binary branch filter, and presents and discusses the experimental results. The report includes a usage manual for the software-deliverable.
Proposal 3: $pq$-Gram Filter for Tree Edit Distance

- Implement the tree edit distance with fanout weighting.
- Implement the $pq$-gram distance (a lower bound of the tree edit distance with fanout weighting).
- Experiments:
  - runtime performance of the tree edit distance with and without filter;
  - effectiveness of the filter on example datasets.

The final report defines the problem, shortly introduces the tree edit distance and the $pq$-gram distance, and presents and discusses the experimental results. The report includes a usage manual for the software-deliverable.
Proposal 4: Traversal String Filter for Tree Edit Distance

- Implement the tree edit distance.
- Implement the traversal string filter (a lower bound of the tree edit distance).
- Experiments:
  - runtime performance of the tree edit distance with and without filter;
  - effectiveness of the filter on example datasets.

The final report defines the problem, shortly introduces the tree edit distance and the traversal string filter, and presents and discusses the experimental results. The report includes a usage manual for the software-deliverable.
Proposal 5: Tree Edit Distance and Windowed $pq$-Gram Distance

- Implement the tree edit distance.
- Implement the windowed $pq$-gram distance.
- Compare the robustness of the windowed $pq$-gram distance and the tree edit distance (with tree sorting) to permutations in the sibling order. Find examples where both approaches do well and where one of them fails.

The final report defines the problem, shortly introduces the tree edit distance and the windowed $pq$-gram distance, compares the robustness to sibling permutations of both approaches, and presents the experimental results. The report includes a usage manual for the software-deliverable.
Implement the following matching algorithms: reverse nearest neighbor, global greedy, stable marriage, and the Hungarian algorithm. The input to all algorithms is a distance matrix, the output are matching (row, column) pairs.

Experiments:

- compare the runtime performance of the four algorithms on distance matrixes of different size;
- compare the effectiveness (recall and precision) of all algorithms on the following distance matrixes for the Bolzano Address Trees: Np3q2.dm, Nw3p2q.dm, Nw5p1q.dm, Nw8p2q.dm.

The final report defines the problem, shortly introduces the matching algorithms, and presents and discusses the experimental results. The report includes a usage manual for the software-deliverable.
Test Data

- Wikipedia Missellings
- Corpora for Missellings
- XML Generator `xmlgen`
- Bolzano Address Trees
- Collection of real world XML
- Your own synthetic data collection
Next Steps...

1. form groups of 2-3 people
2. choose a project proposals
3. send an email to augsten@inf.unibz.it
   - subject: SS Project Proposal X
   - body: "firstname lastname email studentID" for each student in the group
4. you receive user/password of your database as an answer
5. first come – first serve!
Database Connection with JDBC

- **Connection parameters:**
  - host: alcor.inf.unibz.it
  - port: 5433
  - database: ss-0x_2012, $x \in \{1, 2, 3, 4, 5, 6\}$
    - $x$ is the number of your proposal
  - user: equal to database name
  - password: will be assigned to your group

- **Test connection:**
  - `psql -h alcor.inf.unibz.it -p 5433 -d ss-0x_2012`

- **JDBC connection:**
  - jdbc driver:
  - java classpath: `postgresql-9.1-901.jdbc4.jar`