**Progressive Duplicate Detection**

**ABSTRACT:**

Duplicate detection is the process of identifying multiple representations of same real world entities. Today, duplicate detection methods need to process ever larger datasets in ever shorter time: maintaining the quality of a dataset becomes increasingly difficult. We present two novel, progressive duplicate detection algorithms that significantly increase the efficiency of finding duplicates if the execution time is limited: They maximize the gain of the overall process within the time available by reporting most results much earlier than traditional approaches. Comprehensive experiments show that our progressive algorithms can double the efficiency over time of traditional duplicate detection and significantly improve upon related work.

**EXISTING SYSTEM:**

* Much research on duplicate detection, also known as entity resolution and by many other names, focuses on pairselection algorithms that try to maximize recall on the one hand and efficiency on the other hand. The most prominent algorithms in this area are Blocking and the sorted neighborhood method (SNM).
* Xiao et al. proposed a top-k similarity join that uses a special index structure to estimate promising comparison candidates. This approach progressively resolves duplicates and also eases the parameterization problem.
* Pay-As-You-Go Entity Resolution by Whang et al. introduced three kinds of progressive duplicate detection techniques, called “hints”

**DISADVANTAGES OF EXISTING SYSTEM:**

* A user has only limited, maybe unknown time for data cleansing and wants to make best possible use of it. Then, simply start the algorithm and terminate it when needed. The result size will be maximized.
* A user has little knowledge about the given data but still needs to configure the cleansing process.
* A user needs to do the cleaning interactively to, for instance, find good sorting keys by trial and error. Then, run the progressive algorithm repeatedly; each run quickly reports possibly large results.
* All presented hints produce static orders for the comparisons and miss the opportunity to dynamically adjust the comparison order at runtime based on intermediate results.

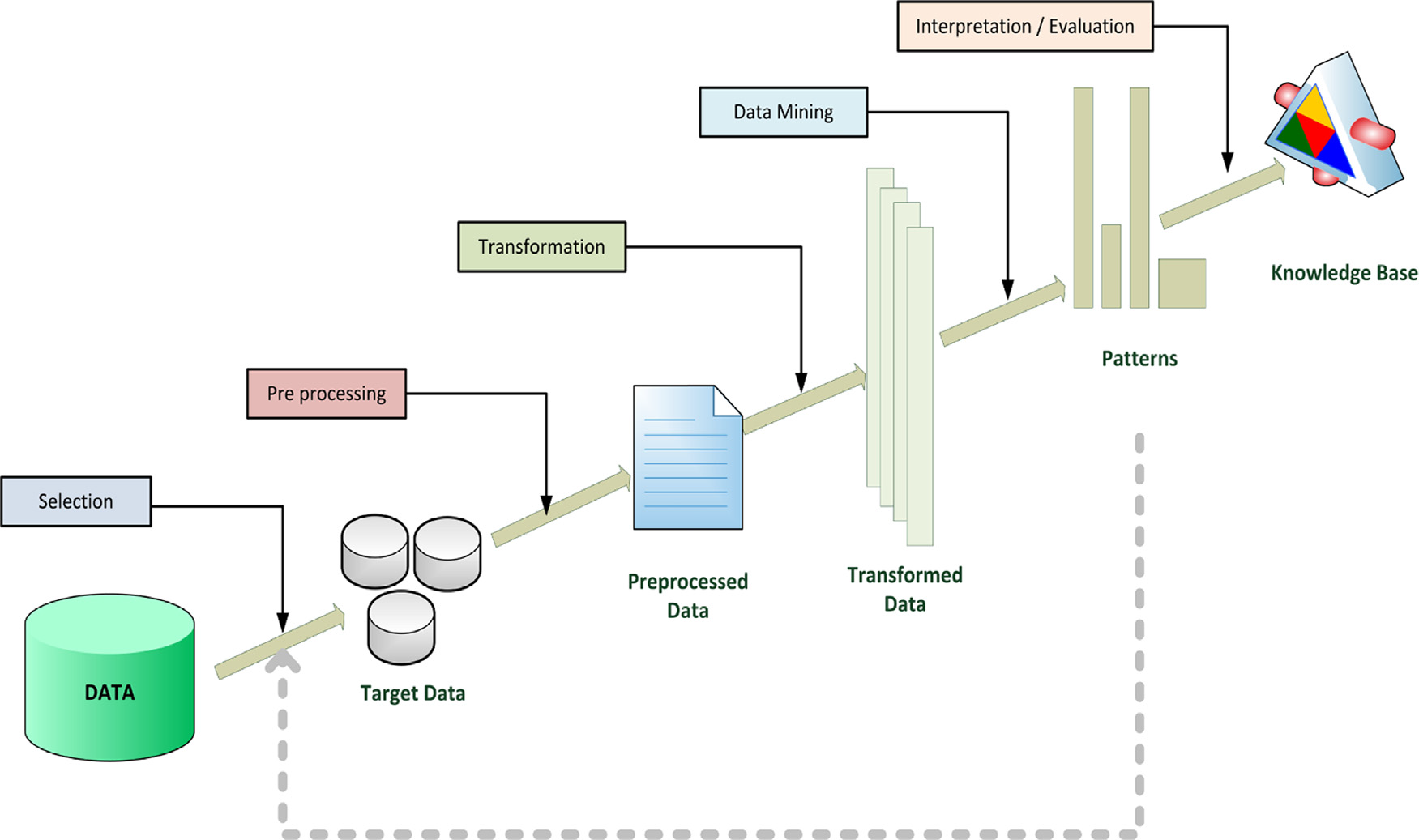
**PROPOSED SYSTEM:**

* In this work, however, we focus on progressive algorithms, which try to report most matches early on, while possibly slightly increasing their overall runtime. To achieve this, they need to estimate the similarity of all comparison candidates in order to compare most promising record pairs first.
* We propose two novel, progressive duplicate detection algorithms namely progressive sorted neighborhood method (PSNM), which performs best on small and almost clean datasets, and progressive blocking (PB), which performs best on large and very dirty datasets. Both enhance the efficiency of duplicate detection even on very large datasets.
* We propose two dynamic progressive duplicate detection algorithms, PSNM and PB, which expose different strengths and outperform current approaches.
* We introduce a concurrent progressive approach for the multi-pass method and adapt an incremental transitive closure algorithm that together forms the first complete progressive duplicate detection workflow.
* We define a novel quality measure for progressive duplicate detection to objectively rank the performance of different approaches.
* We exhaustively evaluate on several real-world datasets testing our own and previous algorithms

**ADVANTAGES OF PROPOSED SYSTEM:**

* Improved early quality
* Same eventual quality
* Our algorithms PSNM and PB dynamically adjust their behavior by automatically choosing optimal parameters, e.g., window sizes, block sizes, and sorting keys, rendering their manual specification superfluous. In this way, we significantly ease the parameterization complexity for duplicate detection in general and contribute to the development of more user interactive applications.

**SYSTEM ARCHITECTURE:**



**Data Separation**

**Duplicate Detection**

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : JAVA
* IDE : Netbeans 7.4
* Database : MYSQL

**REFERENCE:**

Thorsten Papenbrock, Arvid Heise, and Felix Naumann, “Progressive Duplicate Detection”, **IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 5, MAY 2015.**