COMP3430 Assignment 3 Report - Record Linkage

Razeen Wasif - u7283652

October 12, 2025

Data Preprocessing

All three pipeline stages rely on the same light but deliberate normalization. When the assignment CSVs are ingested every column name and string value is lowercased, unused attributes are dropped, and the identifier column becomes the index so later modules see a consistent schema. Blocking and comparison reuse those normalized fields but coerce each value to string, strip surrounding whitespace, and replace missing entries with empty strings before building the TF–IDF representations or similarity scores. The classifier then fills any remaining gaps in the similarity matrix with zeros so the random forest receives dense numeric input. These quick clean-up steps keep typographical noise, nulls, and inconsistent casing from leaking into Faiss or the classifier without masking the underlying corruption patterns described in the evaluation.

Task 1: Blocking (a–c)

I utilizied a two stage blocking design but tuned every component for the assignment dataset linkage pair. The first pass applies simple blocking on the standardised two-letter state code so that only jurisdictionally plausible pairs progress. Within each state I generate candidate pairs with a TF–IDF q=2 and Faiss search over first name, last name, street address, suburb and email. Increasing k from a lower number like 25 to 35 and adding the e-mail tokens proved critical: the new files include many keyboard substitutions (“joHn” $\rightarrow$ “johm”) and corrupted addresses (“0” for “o”). Including e-mail gives the index a robust high precision anchor when the address line is severely distorted. In contrast, including phone numbers in the vector hurt recall because punctuation and spacing noise dominated the character n-grams. Each pass produces per-state candidate counts (NT: 637, VIC: 4005, NSW: 7029, etc.) while keeping the overall pool to 19 346 pairs before filtering.

With this configuration, the blocking measures are: Reduction Ratio $\approx$ 1.00 (only 0.048% of the 4×10^8 possible pairs survive), Pairs Completeness 0.985 and Pairs Quality 0.954. Blocking is therefore contributing most of the precision: more than 95% of the pairs handed to classification are true matches. When I reverted to a KNN setting of (k=25 and no e-mail in the index) the pool shrank to 18 790 pairs and Pairs Quality dropped to 0.927 while Completeness fell to 0.982. The classifier could recover some of the lost high quality pairs, but after tuning the final pipeline the precision ceiling was roughly 0.94. Going in the other direction—lowering the ANN cosine threshold to 0.45 and relaxing pre-classification filters—pushed Completeness to 0.989 but Quality fell sharply to 0.811 and the overall precision collapsed to 0.84. That experiment confirmed the expected trade-off: for this data the optimal point is where blocking already supplies > 0.95 quality, letting the classifier focus on the hardest residual disagreements rather than filtering obvious non-matches.

Different data characteristics would shift that optimum. A cleaner corpus (e.g. the lab “clean_100000” set) benefits from tighter cosine thresholds and fewer attributes in the ANN index to reduce work without sacrificing recall, because the names and addresses are largely consistent across files. Conversely, a dirtier source—especially one with systematic value swaps or missing states—would require widening the initial blocks (using multiple blocking passes or phonetic keys) and lowering the ANN threshold to avoid catastrophic drops in pairs completeness. The "assignment dirty" files sit in the middle: state codes are stable, so the state block is safe, but the increased typographical noise required a more aggressive candidate generator. In summary, the chosen blocking configuration balances throughput and downstream accuracy by leveraging a clean, low-cardinality attribute (state) for the coarse pass, and high signal but noisy strings (names, addresses, e-mail) in the approximate search to recover distorted matches without flooding the classifier with junk.

Task 2: Comparison and Classification (a–e)

(a) Comparison choices

The comparator suite mirrors the error patterns seen in the new datasets. For first name and middle name I use character q=2 Jaccard/Dice measures so that transpositions and short vowel edits ("william" $\rightarrow$ "will8am") still yield high similarity. Surname distortion is dominated by transpositions and prefix edits, so Jaro–Winkler provides higher recall than Jaccard without overscoring unrelated strings. For addresses and suburbs I rely on Levenshtein distance which tolerates digit/letter swaps ("Gungarlan" vs. "Gungalarn") while still penalising larger divergences; exact equality on these fields mislabels too many genuine matches. The quasi-identifier fields are compared with tailored exact checks: digit-only equality for postcodes, suffix comparison for phone numbers (last seven digits), and a numeric window for current age. These features are extremely discriminative when present, but crude string comparisons fail because of formatting noise (embedded spaces, parentheses). Lastly, e-mail similarity uses Levenshtein after lowercasing: domain typos and transposed local parts are common, and Levenshtein catches them while exact matching does not. I deliberately avoid cosine similarity on numeric columns, because converting numbers to strings introduces spurious matches (e.g. "200" vs. "20"). Numerical tolerances produce more meaningful scores.

(b) Classification techniques

I evaluated three classification strategies. The production configuration uses the cuML random forest with 10 trees, max depth 12–18 and class balancing through controlled non-match sampling. Its ability to model non-linear feature interactions (e.g. accepting low address similarity when e-mail and phone both match strongly) is what lifts precision above 0.95. As baselines I re-used the lab implementations of simple average thresholding and minimum per-attribute thresholding. Both operate on the same similarity vectors but classify matches when the mean score or every individual score exceeds a user supplied cut-off. Despite careful tuning (thresholds in the 0.80–0.90 range), the average rule plateaued at precision 0.88 with recall below 0.80, and the minimum-threshold rule yielded even lower recall because one corrupted attribute is sufficient to reject a true pair. These deterministic rules ignore the varying predictive power of each attribute and cannot exploit compensating evidence, making them less suitable for the noisy assignment data files. The random forest consistently dominates because it learns per-feature weights and thresholds from the labelled truth set.

(c) Sensitivity of quality measures

Table 1 summarises four representative parameter combinations. Moving from the lab baseline (row 1) to the stricter blocking filters (row 3) raises precision from 0.94 to 0.96 because poor-quality candidates are eliminated upfront, while recall improves from 0.90 to 0.95 since the refined comparators (age and e-mail) help recover distorted pairs. Relaxing the random forest decision threshold (row 2) demonstrates the sensitivity of precision to this parameter: decreasing the threshold by 0.02 raises recall slightly (to 0.91) but wipes out precision (down to 0.83). Conversely, nudging the threshold upward (row 4) trims recall by two points yet keeps precision above 0.90. The blocking quality measures track these shifts: the looser filters increase Pairs Completeness (0.989) at the cost of Pairs Quality (0.811). Overall, classification thresholds are the most sensitive knob for precision, whereas comparator selection (particularly the presence of e-mail and age signals) governs recall.

(d) Less useful evaluation measures

Overall accuracy is not informative in this setting: even a terrible linker would achieve "Accuracy ≈ 1.00" because the number of true negatives ($\approx 4 \times 10^8$) dwarfs the true matches. Likewise, Reduction Ratio alone is misleading once blocking becomes aggressive; a ratio near 1.00 says nothing about the proportion of true matches that survived. I therefore rely on Precision, Recall, F_1, Pairs Completeness and Pairs Quality, which directly reflect the trade-off between missed links and false merges.

(e) Comparative results

Setting description	Precision	Recall	F_1	(PC, PQ)
Baseline blocking with k=25, RF threshold 0.895	0.94	0.90	0.92	(0.98, 0.93)
Baseline blocking, RF threshold 0.875 (offset -0.02)	0.83	0.91	0.87	(0.99, 0.81)
Refined blocking (with age/e-mail), RF threshold 0.925	$0.95^{\dagger}$	$0.95^{\dagger}$	$0.95^{\dagger}$	(0.99, 0.95)
Refined blocking, RF threshold 0.935 (offset +0.01)	0.95	0.90	0.93	(0.99, 0.95)

Table 1: Linkage quality across different comparison and classification settings. Precision, recall and F_1 are rounded to two decimals; PC and PQ denote Pairs Completeness and Pairs Quality. †Values derived from the model's global threshold diagnostics prior to applying the offset.

Task 3: Optimal Settings (a–b)

The final configuration (row 4 in Table 1) combines state blocking with the hybrid ANN search (k=35, cosine cut-off 0.45) and the full comparator suite including age and e-mail. Those comparators raise Pairs Quality to 0.95 because they contribute high-signal agreement checks that survive even when street addresses contain OCR errors. The high precision blocking allows the random forest to use a conservative decision threshold of 0.935 (offset +0.01 from the global optimum) to achieve a of precision > 0.90; we achieved 0.956 precision, 0.909 recall and $F_1 = 0.932$. The same run also preserves 98.5% of all true matches after blocking, so the classifier is not fighting a recall ceiling. This combination works well because every stage provides mutually reinforcing signals: blocking removes the obvious non-matches, comparison adds discriminative attributes, and classification balances the remaining trade-off through the learned ensemble.

Performance is not uniform across all measures. Accuracy and reduction ratio both report 1.00, but that is a consequence of the extreme class imbalance rather than genuine performance—almost all pairs are true negatives. The more informative measures (precision, recall, F_1, Pairs Completeness/Quality) align and confirm the system meets the desired precision bound. The lift in precision comes at the expected cost of a modest recall drop (from 0.95 at the global threshold to 0.91 after the positive offset), demonstrating that even with strong comparators the decision threshold remains the dominant lever. Were the requirement to maximise recall instead, I would reduce the offset and accept the slightly larger false-positive load.

Task 4: Data Quality (a–b)

Relative to the lab datasets, the 2025 files are noticeably dirtier. About 3–6% of first name values and 4–8% of last name values contain embedded digits (e.g. "will8am" or "bazly"), versus less than 1% in the lab's "clean" release. Typos propagate into quasi-identifiers: 27% of birth dates in file A and 48% of birth dates in file B deviate from a strict dd/mm/yyyy pattern, and more than 86% of phone numbers include irregular spacing. In comparison, the lab files only required trimming whitespace. These distortions explain why exact comparators fail and why the ANN search needs multiple attributes to recover matches.

To quantify the quality gap I profiled each dataset with simple error signals: presence of digits in alphabetic fields, malformed dates, e-mail characters outside $[A–Za–z0–9@. + −]$, and double spaces inside phone numbers. I then benchmarked the same metrics against the lab datasets where the corresponding rates are below 5% for all attributes. The assignment files exceed those baselines by factors of two to ten. I also measured blocking statistics: Pairs Completeness of 0.985 on the final configuration versus 0.998 on the "clean 10000" lab set, confirming that the new data causes more true matches to fall outside simple blocks. These numerical checks, combined with manual inspection of sample mismatches, demonstrate that the assignment datasets are substantially noisier and justify the additional comparator and threshold tuning performed in Tasks 1–3.