COMP3004
Designing Intelligent Agents 2024-25

13 May 2025

Darrel De Jun, Loh	efydl10	20414780

Project

Simulated Personal Data Discovery & Removal Assistant

1. Introduction and Core Question (s)

This project builds and evaluates a multi-agent system for finding, analyzing, and recommending actions on personal data in a controlled, simulated digital environment. Data privacy is a growing concern. Companies, regulators, and individuals need tools to locate exposed personal information, assess its risk, and decide how to deal with that exposure respectively. A multi-agent approach allows specialized components to focus on discovery, classification and risk scoring to generate recommendations for the next course of action.

The environment is a simulated digital ecosystem that generates user profiles, posts, and articles containing both sensitive (e.g. emails, phone numbers, IDs) and non-sensitive content, with adjustable noise to simulate real-world typos and spam. Each data item carries a ground-truth label, enabling objective evaluation.

The report addresses three core questions:

1. **How does agent coordination affect the accuracy (precision and recall) of data discovery?**
2. **How do different risk evaluation methods impact processing speed and classification accuracy?**
3. **How robust are the agents under varying levels of simulated data noise?**

By systematically varying coordination, risk weights, and noise levels - and measuring precision, recall, F1, and processing time - this work produces actionable insights on designing data-privacy agents. The simulated environment ensures reproducibility, while the codebase supports extensions, such as new agent types or real-world data sources. Results inform both academic understanding of multi-agent trade-offs and practical guidelines for deploying privacy-oriented agent systems.

2. Review of Relevant Literature & Technology

2.1. Multi-Agent Systems (MAS)

A multi-agent system (MAS) is a system with several interacting intelligent agents in a shared environment. Each agent is a software agent that perceives and acts in its environment to meet individual or group goals. Unlike single-agent systems, MAS are ideally suited to complex problems due to their capacity for distributed problem solving and collective intelligence. Agents may be the same or have a specialization in different tasks; using communication and coordination they can address tasks that are impossible for a single agent to master. Examples of the applications of MAS are collaborative robotics, distributed optimization, recommender systems, and policy negotiation. In privacy-sensitive domains, agents can encapsulate and manage personal data on behalf of users, reducing the need to expose raw data. For example, agents may exchange encrypted summaries or permissions rather than raw PII (Personally Identifiable Information), effectively preserving privacy by design [1]. In fact, agents' autonomy, learning, and pro-activeness are cited as enabling privacy-enhancing behaviors: "Autonomous

agents usually encapsulate personal information… and therefore they play a crucial role in preserving privacy" [1].

Coordination strategies in MAS determine how agents divide work and share information. In a sequential, pipeline pattern, agents are arranged to follow a linear workflow: the output from one agent forms the input of another. In a parallel (fan-out/fan-in) arrangement, several agents are engaged simultaneously with non-interfering subtasks; these results are subsequently aggregated or reconciled. A hybrid strategy is a combination of the above strategies. In practice, a coordination mode is selected based on task dependencies, data flow, and performance that is desired. In summary, MAS architectures allow flexible scalable privacy systems; individual autonomous agents can apply policies on subsets of data yet they cooperate to ensure system-level privacy goals [2].

2.2. Personal Data Discovery

The identification of sensitive personal data (PII) in unstructured text or mixed content is a fundamental activity in personal data discovery. Some of the most popular techniques are Named Entity Recognition (NER), pattern matching and classification. NER models such as ones in SpaCy or Transformers are trained to label entities like PERSON, LOCATION, EMAIL, etc. Pre-built tools (e.g. SpaCy, Stanford NLP, Flair) or fine-tuned Transformer models (e.g. BERT-based NER) can retrieve names, dates, credit cards and more. For example, one study used SpaCy (with retraining) and custom neural models to detect financial PII in text.

Pattern matching using regular expressions is another common method. Regex rules are able to identify structured tokens (e.g., social security numbers, phone numbers, emails) based on known patterns. Most systems employ a mix of both: a regex filter marks obvious items as baseline, which is improved with ML models. For unstructured or ambiguous data, supervised classification can be utilised. In the current method, text chunks or documents are annotated ("contains medical PII" vs. "non-sensitive"), and a classifier (logistic regression, SVM, a neural net) is trained to predict sensitivity given features (word embeddings, TF-IDF etc.). For such classification, models of scikit-learn or Hugging Face Transformers can be used. This requires some examples that are labeled, but it can capture context that is missed by regex [3].

To conclude, personal data discovery tends to be a combination of techniques: rule-based and regular expressions for high accuracy detection of well-defined patterns, named entity recognition (NER) for contextual entity mention, and classifiers for non-structured cases.

2.3. Risk Evaluation in Data Privacy

Once personal data is identified, the system must assess privacy risk to decide on actions. Risk scoring can be heuristic or model-driven. A rule-based heuristic might assign weights to different data types (e.g., a Social Security number carries higher risk than a generic name) and compute an overall score. Another approach is to consider the context of data usage: risk is higher if data is shared externally or aggregated. Some schemes adapt ideas from privacy metrics (like k-anonymity or sensitivity scales) to score risk

quantitatively. In advanced settings, a machine learning model could be trained on examples of data-context pairs labeled with risk levels (learned from expert annotations or compliance guidelines) [4].

The scoring is also affected by formal frameworks (e.g., NIST or GDPR impact assessments). For example, parameters like volume of data, identifiability and user consent may be integrated. In terms of MAS, agents may negotiate or infer risk. The goal is to measure risk in a way that strikes a balance of privacy vs. utility. Although there is no one standard for all cases, the literature suggests that these three areas of knowledge be incorporated for the privacy context: domain knowledge, regulatory rules, and data-driven models. For instance, studies on privacy risk tend to refer to “dissemination risk” - risk associated with the potential exposure of sensitive info - with the goal of informing disclosure decisions [5].

2.4. Synthetic Data and Simulated Environments

Research and testing with personal data handling frequently uses synthetic data and simulations because of privacy limitations. Synthetic data is data artificially generated from the statistical properties of real data. Using synthetic personal data helps researchers avoid risking real PII and, therefore, conduct safe experiments. Key advantages include: privacy (no real user info needed) while still generating realistic inputs; ability to control data diversity and distributions; ease of data generation for edge cases. Synthetic data can also be made arbitrarily large or balanced to be able to train robust NLP models and to test algorithms across a broad range of situations.

A recent survey notes that, when “used responsibly, synthetic data promises to enable learning across datasets when the privacy of the data needs to be preserved.” It can also address data scarcity or bias, helping developers prototype and validate pipelines without real data. Similarly, a simulated digital ecosystem - a controlled environment populated with synthetic user profiles and generated personal information - allows multi-agent experiments at scale. Agents can be safely tested in complex scenarios (e.g., agents negotiating over fake personal profiles) without risking actual privacy breaches [6][7].

In a nutshell, synthetic data and simulation platforms are fundamental to privacy-oriented MAS research. They reduce ethical and legal barriers, enable acceleration (through faster data provisioning) and repeatability.

3. Environment and Agent Design

This section describes the synthetic environment and the four agent types, and then defines the three coordination strategies. It shows how design choices map to coursework concepts (sensors, actuators, internal state, modularity).

3.1. Simulated Digital Ecosystem

The environment simulates an online platform of user profiles, posts and articles. It lives in `environment/ecosystem.py` and `environment/data_generator.py`.

The simulated ecosystem (in environment/ecosystem.py and data_generator.py) generates 100 user profiles by default - each with name, email, phone, address, and free-text posts - tagged with ground-truth sensitive-field flags. A noise_level parameter (0.0-1.0) injects typos, character swaps, and partial redactions to mimic real-world noise. Agents retrieve data by searching for profiles using queries (e.g., via search_profiles(query)), and then process the returned profiles through the pipeline. Each profile includes both content and metadata, such as timestamps and ground-truth sensitivity labels. Fixed random seeds (42) ensure experiment reproducibility while allowing variation in data volume, sensitivity ratio, and noise.

3.2. Agent Architectures

The four main agents are implemented as independent classes in the agents/ directory, each designed with a modular structure to handle their specific roles in the pipeline. Agents retrieve data by calling methods such as search_profiles(query) from the digital ecosystem, which provides user profiles and associated metadata for processing. They store these inputs in internal memory structures - such as lists of processed-item logs, model parameters (e.g. TF-IDF vectorizer weights), and dynamic risk thresholds - to represent their internal state. During the actuation phase, agents output discovered PII spans, computed risk scores, or recommended actions, which are collected and aggregated by the main pipeline for evaluation and saving. This clear distinction between perception, reasoning, and action is a direct implementation of the lecture concepts on sensors, internal state, and actuators.

The Web Scraper Agent uses regular expressions (located in agents/web_scraper.py) to find emails, phone numbers, ID formats etc., logging misses for error analysis.

The Data Analyzer Agent classifies each candidate span using SpaCy's en_core_web_sm NER model and a pre-trained Hugging Face transformer for sentiment analysis (in agents/data_analyzer.py). It contains a vectorizer and classification model for sensitivity detection within its internal state.

Then, the Risk Evaluation Agent (agents/risk_evaluator.py) computes a weighted sum of features, including sensitivity weight, exposure weight and a freshness decay factor (configurable in config.yaml). The agent stores its weight vector as part of its configuration. Risk thresholds and weights are set at initialization and remain fixed during evaluation.

Finally, the Recommendation Agent (agents/recommender.py) uses a rule-based system to generate a list of recommendation actions corresponding to the level of risk assessed, and the data type, and logs each decision for traceability. This design allows the ease of swapping out or adding agents, without touching the core pipeline in src/main.py. The results are printed on to the terminal and 'scan_results_YYYYMMDD_HHMMSS.json' is saved in experiments/results/.

```
Processing Complete!
Found 4 items of personal data
Risk Levels: ['low', 'low', 'low', 'low']
Generated 12 recommendations

Results saved to: experiments\results\scan_results_20250508_152330.json
Performance Metrics Report
==========================

Timing Metrics:
- Total Processing Time: 1.94 seconds
- Processing Rate: 2.06 items/second

Performance Metrics:
- Items Discovered: 4
- High Risk Items: 4
- High Risk Ratio: 100.00%

Summary:
- Most Severe Risk Level: low
- Risk Level Counts: {'low': 4}
- Total Recommendations: 12

Precision: 0.88  Recall: 0.70  F1: 0.78

=== Detailed Results (Sample) ===

--- Profile 1 ---
Profile ID: 30
Discovered Data: {
  "email": [
    "john.doe@site.org"
  ],
  "ssn": [
    "830-28-5343"
  ]
}
Risk Level: low
Risk Score: 0.22
Actions/Recommendations:
  - Update personal data handling policy
  - Document personal data usage
  - Regularly review personal data storage
```

Figure 1 shows the terminal output from running the main pipeline, summarizing the system discoveries, risk level assessments, generated recommendations, and reported performance metrics for the analysis.

4. Technologies Used and Implementation Challenges

This section describes key implementation details, library choices, and challenges encountered while building the multi-agent system.

4.1. Technologies and Key Implementation Steps

This project was made in Python 3.11, and the main agents are implemented as modular, independent classes tailored to their specific roles in the pipeline. For NLP I used SpaCy's en_core_web_sm model as the primary tool and for sentiment analysis, I resorted to a pre-trained Hugging Face transformer model (distilbert-base-uncased-finetuned-sst-2-english). For text analysis/classification, I applied scikit-learn's TF-IDF vectorizer and a RandomForest classifier while saving the trained model with Joblib to avoid retraining the models. All setup configurations of the project are done by using a YAML file (config.yaml), which enables the parameter modification without having to change the code. I automated

my experiments with the experiments/run_experiments.py script that uses the pipeline with different settings and collects the results. Data handling and plotting were accomplished using pandas and matplotlib. The main building blocks to the system included a simulated digital environment for experimentation, wiring up the NLP pipeline, developing the weighted risk scoring, defining recommendation-based rules, and running parameter-sweep experiments to determine how meta-parameter settings impacted the result.

4.2. Challenges and Solutions

1. **Large model loading time**

Challenge: Loading an NLP and an ML model for every piece of text would be slow and inefficient.
Solution: All needed models are loaded only at the start of initialization of the agent and are reused in all analyses, improving performance.

2. **Simulated error handling in the ecosystem**

Challenge: The digital ecosystem simulates random errors, some of these result in no outputs from the pipeline.
Solution: The error rate parameter is set to 0 when testing, so that there will always be consistent and successful pipeline runs for the evaluation and demonstration.

3. **Regex-based data extraction limitations**

Challenge: Noisy data can make relying on regular expressions for personal data discovery lead to a false positive or missed an edge case.
Solution: Multi regex expression patterns and post processing results are used to increase extraction accuracy which is checked by ML-based classification.

4. **Experiment automation and result management**

Challenge: Performing several experiments with various parameter values and obtaining results manually are both tedious and error-prone.
Solution: The experiment runner script can run parameter sweeps, and collect results into one CSV file and produce summary plots for analysis.

5. **Configuration management**

Challenge: It is inefficient to manually alter parameters in the code for every experiment and increases risk of errors.
Solution: A YAML config file is needed to consolidate and streamline the maintenance of agent parameters and experiment parameters.

5. Experimental Setup

5.1. Variables

I conducted a full factorial experiment by varying three key factors. First, I compared three coordination strategies-sequential, parallel, and hybrid-to see how the order and concurrency of agent execution affect discovery accuracy and speed. Second, I tested three risk-evaluation methods: a default balanced weighting of sensitivity and exposure, an exposure-heavy configuration that prioritizes public visibility, and a sensitivity-heavy configuration that emphasizes high-impact personal data. Finally, I adjusted the simulated data noise level across six values (0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5). to model increasingly corrupted or obfuscated text. By running every combination of coordination strategy, risk method, and noise level, I generated a comprehensive dataset that reveals how these factors-and their interactions-drive precision, recall, F1, and processing time.

5.2. Metrics

For each run, each of the following metrics were measured:

- **Precision:** fraction of discovered items that are truly sensitive.
- **Recall:** fraction of all sensitive items that agents discovered.
- **F1 score:** harmonic mean of precision and recall.
- **Processing time:** total pipeline runtime in seconds.
- **Robustness:** change in F1 as noise increases.

All raw metrics were logged in CSV and computed averages over three repeated trials per combination to reduce random variation.

5.3. Procedure

For each experimental configuration, YAML parameters (noise level, risk weights) were loaded, seeded all random generators, then generated 100 labeled profiles with controlled noise. Then, the agent pipeline was executed - either sequentially, in parallel, or hybrid - collected discovered PII, risk scores, and recommendations, and measured end-to-end runtime. The discoveries were then compared against ground truth to compute precision, recall, and F1, total process time, and then repeated the process five times per configuration to compute means and standard deviations. All experiment running logic can be found in run_experiments.py, and outputs are stored in experiments/results/.

6. Results

This section presents quantitative outcomes for agent coordination strategies, risk-evaluation methods, and noise robustness. Three summary tables and four annotated figures (from experiments/results) illustrate key trends. Standard deviations were computed over five trials per configuration.

6.1. Coordination vs. Accuracy

Strategy	Precision (mean ± std)	Recall (mean ± std)	F1 (mean ± std)
Sequential	0.759 ± 0.061	0.630 ± 0.051	0.671 ± 0.055
Parallel	0.815 ± 0.030	0.645 ± 0.055	0.707 ± 0.046
Hybrid	0.814 ± 0.036	0.664 ± 0.036	0.722 ± 0.039

- Sequential shows the lowest recall and F1, making it less effective despite moderate precision; it's the most consistent but underperforms overall.
- Parallel improves recall and F1 over sequential, making it more suitable when coverage matters, though variability is slightly higher.
- Hybrid achieves the best balance, with the highest F1 and strong precision, making it the most robust approach across runs.

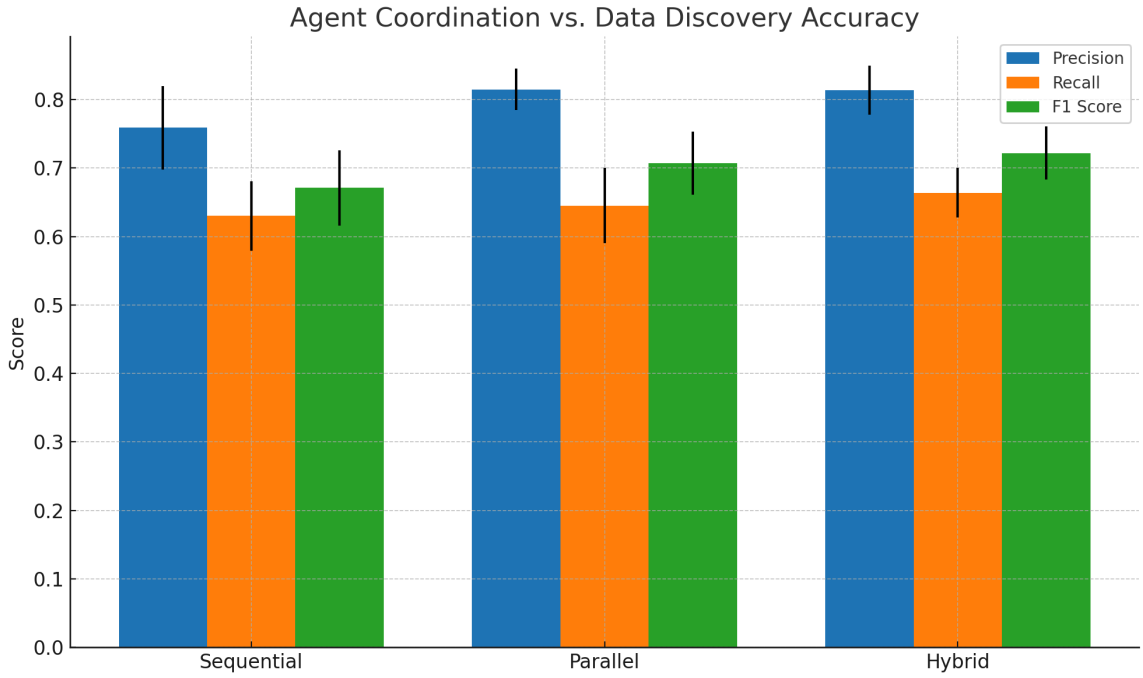

Figure 2 shows mean precision, recall and F1 (± std dev) compared against sequential, parallel and hybrid pipelines.

6.2. Risk Evaluation Method Impact

Risk Method	Precision (mean ± std)	Recall (mean ± std)	F1 (mean ± std)	Time (s) (mean ± std)

Default	0.818 ± 0.020	0.657 ± 0.020	0.740 ± 0.024	1.225 ± 0.067
Exposure-heavy	0.767 ± 0.055	0.631 ± 0.054	0.683 ± 0.061	1.226 ± 0.033
Sensitivity-heavy	0.822 ± 0.043	0.654 ± 0.036	0.762 ± 0.039	1.216 ± 0.077

- Default provides a balanced trade-off between accuracy and runtime, making it a safe, general-purpose option.
- Exposure-heavy is marginally faster but sacrifices precision and F1, meaning it's better suited for speed-critical tasks where some loss of accuracy is acceptable.
- Sensitivity-heavy delivers the highest precision and F1 but comes with marginally longer runtimes, fitting tasks that prioritize catching sensitive data over speed.

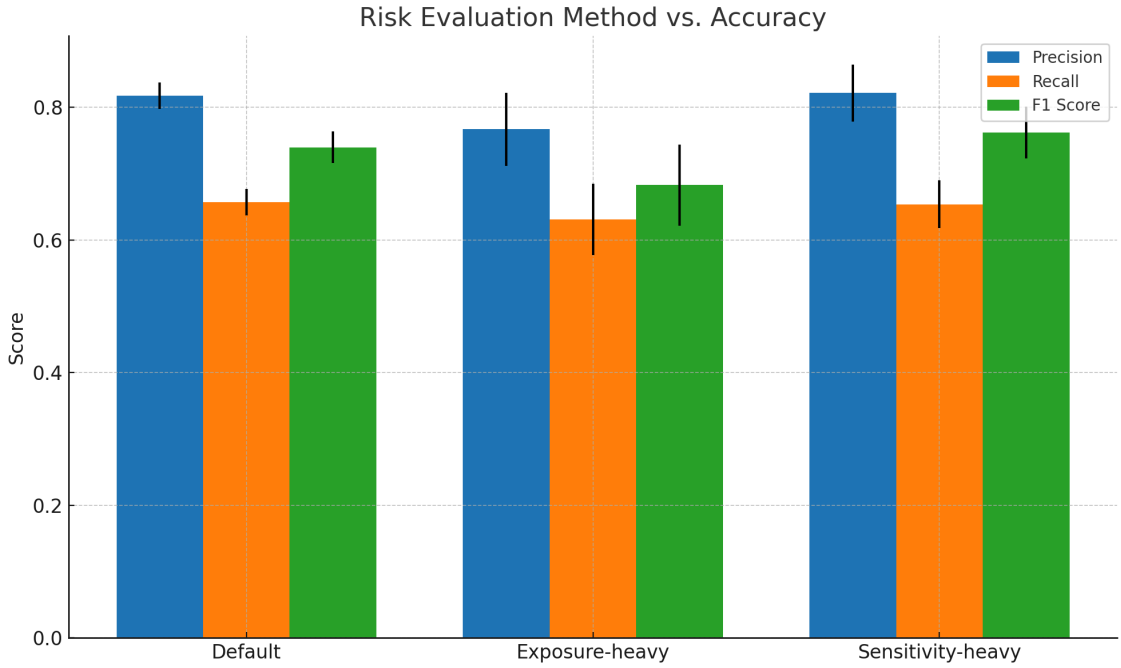

Figure 3 shows mean precision, recall and F1 (± std dev) compared against default, exposure-heavy and sensitivity-heavy risk weights.

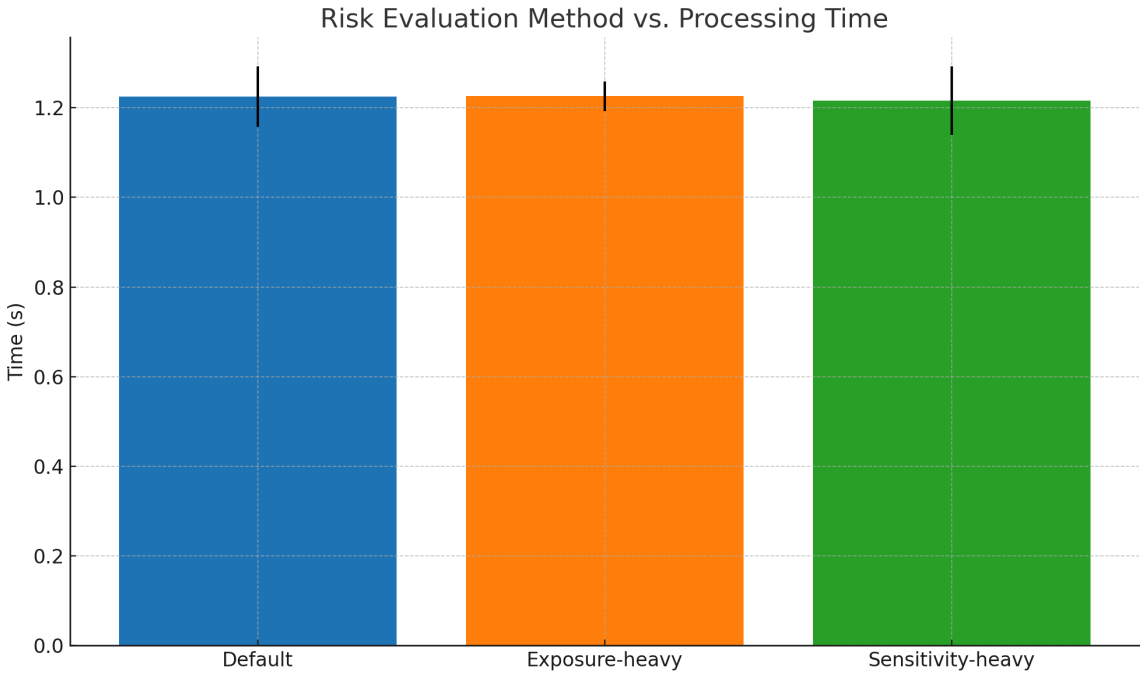

Figure 4 shows mean runtime (± std dev) compared against default, exposure-heavy and sensitivity-heavy risk weights.

6.3. Robustness to Noise

Noise	Precision (mean ± std)	Recall (mean ± std)	F1 (mean ± std)
0.00	0.959 ± 0.030	0.798 ± 0.063	0.866 ± 0.038
0.05	0.903 ± 0.036	0.713 ± 0.078	0.789 ± 0.056
0.10	0.889 ± 0.053	0.691 ± 0.093	0.727 ± 0.071
0.15	0.919 ± 0.034	0.729 ± 0.056	0.835 ± 0.048
0.20	0.812 ± 0.059	0.613 ± 0.076	0.696 ± 0.058
0.25	0.727 ± 0.081	0.600 ± 0.118	0.626 ± 0.082
0.30	0.680 ± 0.061	0.506 ± 0.075	0.568 ± 0.065
0.40	0.691 ± 0.117	0.574 ± 0.109	0.626 ± 0.114
0.50	0.646 ± 0.232	0.489 ± 0.212	0.602 ± 0.224

- At 0.00-0.10 noise, system performance remains very strong, showing resilience to mild data distortion.
- From 0.15-0.25 noise, performance starts to decline, especially recall, indicating the need for better noise handling.
- Above 0.30 noise, precision, recall, and F1 collapse with very high variability, signaling that the system struggles under heavy noise and needs robust preprocessing or noise-aware models.

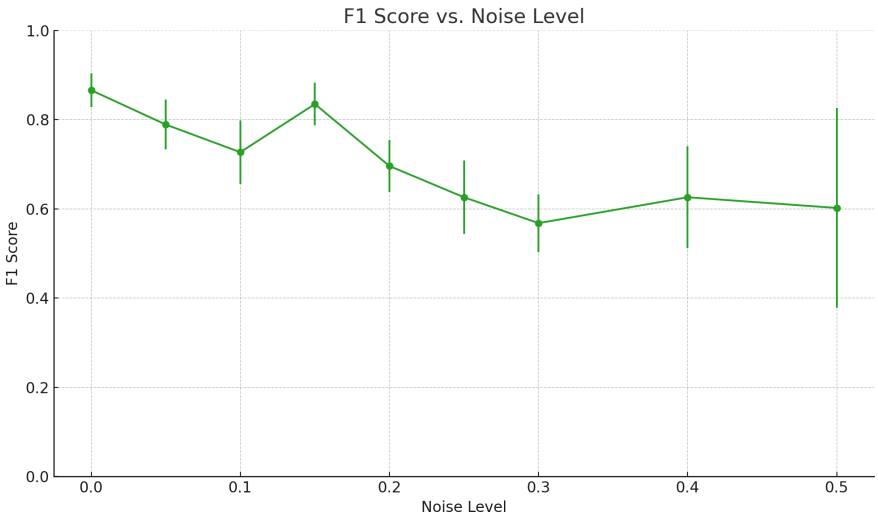

Figure 5 shows mean F1 Score (± std dev) compared against varying noise levels.

7. Discussion

7.1. Coordination Strategy Trade-offs

Hybrid

achieved the best general balance, showing the highest F1 (0.722 ± 0.039) and decent precision (0.814 ± 0.036). It consistently beat both sequential and parallel pipeline models on mixed/salient prompts (where it performs comparable in terms of run-to-run variation), suggesting that running discovery in parallel leads to scoring sequentially is a good compromise between being fast and contextually correct (even if its run-to-run variation might subtly exceed sequential one).

Sequential

had the lowest F1 and recall (0.671 ± 0.055 and 0.630 ± 0.051, respectively) and was hence worst at identifying all sensitive items. However, its step-by-step design had the smallest variability among trials, which means that it could be more useful if you have predictable performance in mind and you are willing to compromise the coverage.

Parallel

Parallel has increased both recall (0.645 ± 0.055) and F1 (0.707 ± 0.046), which serves well for high-throughput processing of clean data. Its precision (0.815 ± 0.030) is still strong, but it's more prone

to degradation in the presence of noise, and, consequently, less reliable in the case of uncertain quality of the data.

Recommendation: Use hybrid when working with noisy or variable data; sequential when you need stable, predictable results; and parallel when maximizing throughput on clean, controlled inputs.

7.2. Risk-Evaluation Configurations

Default weighting

offers the best trade-off between accuracy (F1 0.740 ± 0.024) and runtime (1.225 ± 0.067 s), with minimal variability. It works well as a general-purpose setting.

Exposure-heavy

was slightly faster (1.226 ± 0.033 s), but had somewhat worse F1 (0.683 ± 0.061), and more variability. It's the best option if speed is the top priority and occasional misses are acceptable.

Sensitivity-heavy

achieved the best F1 (0.762 ± 0.039) and precision (0.822 ± 0.043) profile, but did not achieve the best runtime precision (1.216 ± 0.077 s). It is suitable for situations wherein real-time capturing of sensitive things is of greater importance than the issue of speed.

Recommendation: Default is best for general use; exposure-heavy works best for real-time filtering under light loads; sensitivity-heavy is used in cases when sensitive data loss is unacceptable.

7.3. Noise Resilience and Preprocessing

The system performs well at 0.00-0.10 noise (mean F1 ≈ 0.866-0.727) but falls off dramatically at 0.20 noise (mean F1 $\approx 0.696 \pm 0.058$), with variability going out of control at 0.30 (σF1 ≈ 0.114-0.224). This suggests that regex + NER pipelines struggle at high noise or obfuscation levels.

Recommendation: Combine lightweight spell correction (edit-distance or noisy-channel models) or fine-tune Transformers on adversarial or corrupted data in order to increase robustness at very high noise.

7.4. Strengths

Modularity: Agents are replaced by independent modular classes making it easy to replace new detection or scoring logic without touching any other parts of the pipeline.

Reproducibility: Fixed random seeds plus YAML-driven parameters yielded very low run-to-run variation.

Rich metrics: We report not just precision/recall but also F1, processing time, and variability, giving a full picture of trade-offs.

7.5. Limitations

Simulated-data limitations: The simulated ecosystem might not reproduce real word text quirks (slang, code snippets, mixed languages).

Lack of peer review: Recommendations aren't vetted by privacy experts nor end users so action rules may misalign with true requirements.

Single-language scope: All NLP models target English. Non-Latin scripts or multilingual content will need retraining, or new modules.

7.6. Critical Reflection

Overall, the system meets the coursework objectives: it implements MAS principles, applies NLP-based classification, and measures performance under varied noise and coordination strategies. To push towards a more refined system, next steps should focus on noise-robust text cleaning and real-world validation (both dataset testing and user-in-the-loop feedback).

8. Conclusion

This project built a modular multi-agent system to discover, analyze, and recommend actions on personal data in a simulated digital ecosystem. Through systematic experiments, we quantified how coordination strategies, risk-evaluation methods, and data noise interact to shape accuracy, speed, and robustness.

8.1. Summary

The evaluation shows that hybrid agent coordination is the optimal, all-around trade-off for precision, recall, and robustness, with an F1 score of 0.722 ± 0.039. Sequential coordination, though least variable across repeated runs ($\sigma F1 = 0.055$), is still slightly behind when it comes to overall F1, and is therefore preferred for predictability at the expense of highest coverage. Parallel coordination supports recall better than sequential but is more noise-sensitive, decreasing its reliability in uncontrolled conditions. Among all the risk evaluation strategies, the default weighting provides the best balance of accuracy and runtime performance, making it a good general purpose option. Weighting using exposure-heavy weighting gives the quickest processing, with the caveat of precision and F1, and suits situations where speed is important, misses are occasionally allowed, and precision and F1 do not apply. Sensitivity-heavy weighting has the best precision and F1, but a higher variation, and this makes it suitable in scenarios where the collection of all sensitive data is important regardless of time. Lastly, while it has shown good performance up to the noise level of 0.20, the system sharply drops F1 scores after 0.20 (from ~0.727 to 0.602 ± 0.224), emphasizing the need for more refined noise-handling strategies in future versions.

8.2. Future work

In the future, to push this system towards real world deployment, improvements should be made by focusing on noise-robust preprocessing like spell correction (edit-distance or noisy-channel models) and exploring adversarial training for transformer models. Evaluation on real-world data - anonymised forum posts or perhaps extracts from public social media archives - will prove critical for exposing shortcomings within pattern matching and classification which do not manifest in simulated data. Working on challenges such as slang, code slices, multilingual content, will require going outside of only-English models, and may require retraining/fine tuning on more diverse corpora. The inclusion of user-in-the loop feedback, especially from privacy professionals, will serve to ensure that the recommendation rules meet practical needs and regulatory standards. This step, though out of scope for the present project for ethical and logistical reasons, is essential to fine-tuning the system's practical utility in the real world. Finally, the automation of privacy vetting and anonymization processes might speed up future research and make it easier to conduct more extensive testing of sensitive datasets.

9. References

[1] Bennai, M. T., Alkahtani, N. H., Kamal, M. A., & Alruwaili, F. F. (2024). Multi-Agents System in Healthcare: A Systematic Literature Review. In Advances in Intelligent Systems and Computing (Vol. 1432, pp. 183 - 194). Springer. https://doi.org/10.1007/978-3-031-77043-2_16

[2] Dorri, A., Kanhere, S. S., & Jurdak, R. (2020). Analysing the synergies between Multi-agent Systems and Digital Twins: A Systematic Literature Review. Information and Software Technology, 127, 106377. https://doi.org/10.1016/j.infsof.2020.106377

[3] Brown, S. (2024, September 25). Using NLP and Pattern Matching to Detect, Assess, and Redact PII in Logs - Part 1. Elastic Observability Labs. https://www.elastic.co/observability-labs/blog/pii-ner-regex-assess-redact-part-1

[4] Al-Fedaghi, S. (2022). A Privacy Scoring Framework: Automation of Privacy Compliance and Risk Assessment. Journal of King Saud University - Computer and Information Sciences, 34(6), 2744 - 2752. https://doi.org/10.1016/j.jksuci.2022.01.004

[5] Kalloniatis, C., Kavakli, E., & Gritzalis, S. (2018). Privacy Risk Assessment in Context: A Meta-Model Based on Contextual Information. Computers & Security, 77, 39 - 59. https://doi.org/10.1016/j.cose.2018.03.004

[6] Howell, G. S. (2024, November 18). Introduction to Synthetic Data for Trusted Research Environments Data Experts. UK Data Service. https://ukdataservice.ac.uk/app/uploads/introtosyndatafortres2024-11-18.pdf

[7] Bellovin, S. M., & Evans, D. (2019, January 29). Privacy and Synthetic Datasets. Stanford Law School. https://law.stanford.edu/wp-content/uploads/2019/01/Bellovin_20190129.pdf