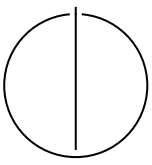
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Improving and Benchmarking Privacy in RAG

Kevin Muyuan Zhou



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

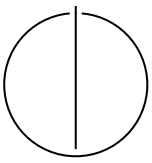
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Improving and Benchmarking Privacy in RAG

Titel der Abschlussarbeit

Author:	Kevin Muyuan Zhou
Examiner:	Supervisor
Supervisor:	Advisor
Submission Date:	19.09.2025



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 19.09.2025 Kevin Muyuan Zhou

Acknowledgments

HI

Abstract

Hi

Contents

1 Introduction

1.1 Section

Citation test [Lam94].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}, \ac{TUM}` ⇒ Technical University of Munich (TUM), TUM

For more details, see the documentation of the `acronym` package[1].

1.1.1 Subsection

See Table 1.1, Figure 1.1, Figure 1.2, Figure 1.3.

Table 1.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

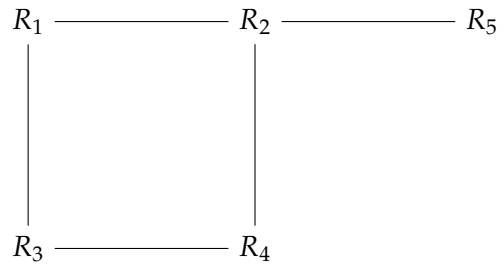


Figure 1.1: An example for a simple drawing.

[1] `https://ctan.org/pkg/acronym`

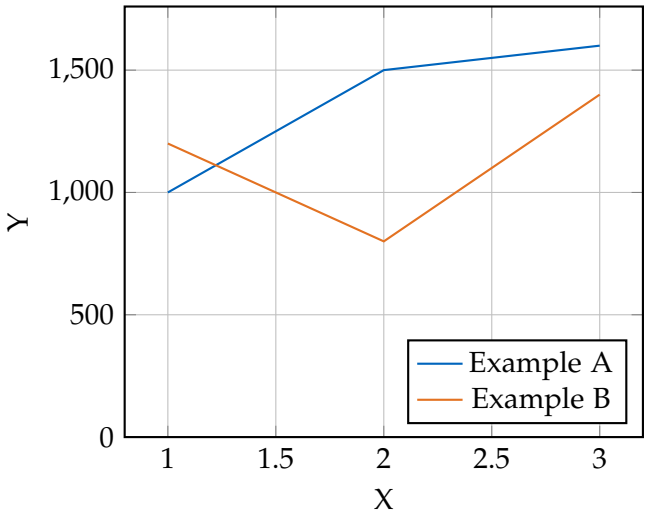


Figure 1.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 1.3: An example for a source code listing.

2 Background and Threat Model

This section relies heavily on the definition presented in [Exploring Privacy issues in RAG].

2.1 Standard RAG Pipeline

The Retrieval-Augmented Generation (RAG) system consists of a large language model M, a retrieval dataset D and a retriever R. To answer a query q, the retriever R fetches the k most relevant documents from D. The relevance of a document to a query is typically measured by calculating the similarity or distance between the query embedding e_q and the document embedding e_d. Formally:

$$R(q, D) = \{d_1, d_2, ..., d_k\} \text{ with } dist(e_q, e_{d_i}) \text{ for } i \in \{1...k\} \text{ in the top } k$$

Common similarity measures include cosine or L2. Given the k-most relevant documents the answer a is generated using the language model M by combining the retrieved documents with the query.

$$a = M(R(q, D)||q)$$

2.2 Taxonomies

2.3 Threat Model

For the adversary we assume a black-box scenario, where the attacker's access to the model is limited to API queries. Therefore the attacks are limited to carefully designed queries accumulate responses over multiple sessions.

3 Literature Review

bb

4 Approach

4.1 Overview

This chapter introduces a risk-aware preprocessing pipeline that detects, quantifies and mitigates the risk of cross-document linkage in unstructured texts. By combining the extracted potentially-identifying entities from the text with their relevance, corpus-wide uniqueness and risk weights, a document-linkage graph is created. Using this graph, critical entities and connections are identified and selectively blurred.
The approach attempts to balance privacy-preservation and downstream task performance by creating the smallest set of substitutions necessary to reduce both per-document as well as cross-document linkage risk below predefined thresholds.
The presented implementation targets a health-insurance setting with an adapted entity schema and prompts. But the method itself is domain-agnostic, with the strength of the privacy-preservation being configurable via hyperparameters like risk thresholds and chain lengths.
Our proposed approach focusses on modifying the retrieval dataset D, while leaving the Retrieval-Augmented-Generation untouched to avoid additional computational costs during inference. It has to be executed once before the documents are inserted into the RAG-Systems database for retrieval.

4.2 Pipeline

4.2.1 Data and Normalization

The documents are ingested from a folder of JSON Files, each containing the fields: id, metadata and content. The content is normalized to stabilize matching and hashing. A unique identifier based on the provided and its normalized content using the md5 hash is assigned to avoid collisions and clear logging.

$$\text{document\_id} = \text{id}_{doc} || \text{md5(normalized\_content)}$$

The normalization is kept minimal here, converting the text to lowercase to ease matching during later steps. Additional domain-specific normalizations, like splitting

or cleaning of datasets containing fix-format data, can be applied here, but are not necessary for the pipeline to function. All documents are gathered in a global list of `Document` objects for later access.

4.2.2 Entity Schema and Extraction

This section presents an entity type system tailored to the health-insurance setting and a two-stage entity extraction process that uses an LLM to first extract local patient-related entities, then performs a secondary extraction run to discover cross-document linkage.

Entity Schema (Insurance)

For the given setting, the following entity-weight-schema is introduced. It is designed to capture direct- as well as quasi-identifiers and assigns a weight w_{type} to each type, representing the severity in case of leakage. The weight is defined as $w_{type} \in [0, 1]$, with higher values indicating higher severity.

Entity Type	Risk Weight
NAME	1.00
PATIENT_ID	0.95
ADDRESS	0.90
PHONE_NUMBER	0.85
MEDICAL_CONDITION	0.85
EMAIL	0.80
NON_PERSONAL_ID	0.80
UNIQUE_FACT	0.78
BIRTHDATE	0.75
TREATMENT	0.72
INDIRECT_IDENTIFIER	0.70
PROVIDER	0.65
EVENT_DATE	0.60
AGE	0.55
LOCATION	0.55
EVENT	0.50
DEMOGRAPHIC	0.35

Table 4.1: Entity types sorted by descending risk weight in an insurance context.

Output format and normalization

Each extraction stage returns a JSON of the following form. The `original_value` represents the verbatim value of the entity in the document while the `normalized_value` is used to unify different representations of the same concept or entity during later stepts. For example: "12 April 2019" and "12/04/19" would both be normalized to "12/04/2019". `entity_type` is one of the entity types defined in 4.2.2 and `relevance` $\in [0, 1]$ is set by the LLM, indicating how useful an entity is for re-identification.

```
{
    "entities": [
        [original_value, normalized_value, entity_type, relevance],
        # additional entities ...
    ]
}
```

Listing 4.1: Response JSON schema

Each extracted entity is given a unique identifier and is parsed into a `GlobalEntity` object and a `EntityInDoc` object. The former object stores all global information like: a list of original values, the normalized value, the type and documents this entity appears in, while the latter is stored on a per-document basis, including only the id and the relevance of an entity for one document.

$$\text{entity\_id} = \text{md5}(\text{normalized\_value}||\text{entity\_type})$$

Local Entity Extraction

Each extraction is performed on a per-document basis. For each request, the model receives the normalized content along with an prompt that specifies the entity schema described in Section 4.1. The prompt instructs the LLM to extract relevant entities for patient-identification and specifies extraction and normalization rules. The exact phrasing of the prompt can be found in Appendix **??**.

Context-Aware Entity Extraction

The second extraction step is based on the results of local extraction. Here the model receives the document content together with a set of previously extracted normalized entities, referred to as `existing_entities`. More specifically, `existing_entities` consists of a list of (`normalized_value`, `entity_type`) tuples. The previously set relevance of an entity is purposely left out to allow the LLM to set a new, unbiased relevance score.

The prompt is modified to encourage the search for matches between the `existing_entities` and the document content. This provides context for the extraction, allowing the LLM to uncover entities that were not extracted in the first pass, but gained potential through cross-document recurrence. For example, a study mentioning a specific demographic (e.g. "construction workers") may not initially be considered sensitive. However, if another document links the same demographic to a medical condition, the combined information becomes more relevant for potential re-identification. By leveraging this context, the LLM becomes more sensitive to fragmented information and is able to create previously unseen links between documents.

Issue: Length of Passed Context While the approach is effective for smaller numbers of documents, the performance degrades when the list of `existing_entities` becomes too long. Direct identifiers like names are extracted during the local entity extraction, but not during the second extraction, despite the given context. This problem occurs despite the model having a sufficient context length. To mitigate this, the following filtering strategies were explored

Heuristic Reduction via Uniqueness and Relevance Each entity is assigned a uniqueness score (Details regarding calculation in 4.2.3). The product of the uniqueness and the maximal relevance across all documents of an entity make up the filter score:

$$\texttt{filter\_score} = \texttt{max\_relevance} \cdot \texttt{uniqueness\_score}.$$

Entities with a filter score below a chosen threshold (e.g. 0.3) are excluded from the context. The rationale for this design is twofold:

- entities that occur frequently across documents are already sufficiently "visible" to be extracted without additional context, and
- entities with low relevance are typically too generic (e.g., broad locations such as "Massachusetts" or vague dates such as "last year"), and thus rarely contribute meaningful information.

This strategy ensures that only entities with sufficient uniqueness and relevance are added to the context for the second extraction step.

Filtering by Entity Type Some high-risk entity types, such as direct identifiers (e.g., names, emails, addresses) are inherently sensitive regardless of the context. Therefore, including them in `existing_entities` provides little value for identifying previously overlooked entities. However, since these direct identifiers only make up a small fraction of all extracted entities, omitting them does not significantly reduce the length of the passed context.

Limiting the Number of Entities Another strategy is to impose a hard limit on the number of entities included in the context. For instance, we could select only the top-k entities, where k elements are chosen using one of the following strategies:

- **Most relevant entities:** prioritizes high-utility entities but risks redundancy, as these are already sensitive without additional context
- **Least relevant entities:** may uncover context-dependent links, but often includes overly generic entities that add create generic links between documents
- **Most recent entities:** assumes temporal locality of list of processed documents, e.g. documents processed after another are related.

For the final implementation, a combination of heuristic reduction and type-based filtering was employed to balance context length with utility.

4.2.3 Privacy Analysis

This section formalizes how, given the entities and their relevance, the pipeline quantifies the privacy risk on a single- and multi-document level. Potential privacy violations are detected and marked, allowing for targeted blurring in later steps.

Uniqueness

Uniqueness is a global property, that captures how rare an enityt is with respect to the entire document collection. Let N be the number of documents and let an entity e occur in $freq_e$ distinct documents. We define an IDF-inspired uniqueness score $u(e) \in [0, 1]$:

$$u(e) = \frac{log(\frac{N+1}{freq_e})}{log(N+1)}$$

The score assigns the maxiumum uniqueness (≈ 1) to entities that appear only once in the corpus and decays smoothly for more frequent entities. Reason for this formula is, that unique or near-unique entities (e.g. patient names, rare diseases) are much more useful for re-identification than common entities (e.g. a countries name). Using this normalized IDF function creates a simple and efficiently computable measure.

Document Risk

To calculate a single document risk score $R(d)$, we first need to calculate the *per-entity contribution* $c(e, d)$ for each entity in the document.

Entity Contribution Each extracted entity consists of three scalar factors:

- `relevance` $\in [0, 1]$: a document-specific score set by the LLM, indicating the value of an entity regarding re-identification
- $u(e) \in [0, 1]$: the global uniqueness defined above,
- $w_{type} \in [0, 1]$: a fixed risk weight that depends on the entity type t (see Table 4.1).

We set the *per-entity contribution* to:

$$c(e, d) \;=\; \texttt{relevance}(e, d) \cdot u(e) \cdot w_{type(e)}.$$

Intuitively, an entity has a large contribution if it's highly relevant in a document, globally unique and describes sensitive attribute (e.g. NAME, EMAIL) of a person.

Accumulation of Entity Contributions Given the *per-entity contributions* $c(e, d)$ of all entities e in document d, we aggregate them into the final document risk score $R(d)$. We propose the following complement-of-product formulation:

$$R(d) \;=\; 1 - \prod_{i=1}^{m} (1 - c_i).$$

This formula treats each entities contribution as independent for re-identification and returns a bounded risk-score $R(d) \in [0, 1]$. Additionally, this formula has multiple desirable properties:

- a single large contribution c_i pushes $R(d)$ close to 1
- high-risk entities cannot be dilluted by multiple low-risk (e.g. event dates, demographics) entities
- multiple medium to low risk entities accumulate, but with diminishing returns (e.g. 5 locations increase risk, but not linearly)

Document Graph

To model cross-document linkage, we construct an undirected document graph $G = (D, E)$ where each node $d \in D$ represents one document. An edge $(d_i, d_j) \in E$ exists when both documents d_i and d_j share at least one entity. Each edge stores:

- `via`(e): the set of shared entity IDs connecting both documents,

- `edge_strength` $\in [0,1]$: a computed value describing the strength of the link between documents

The `edge_strength` is computed like the document risk, accumulating the *per-entity contribution* of shared entities. We handle the potential difference in `relevance` by choosing the higher relevance for the calculation. This results in the following formula:

$$s_e(d_i, d_j) \;=\; max(\texttt{relevance}(e, d_i), \texttt{relevance}(e, d_j)) \cdot u(e) \cdot w_{type(e)}.$$

The accumulation step is identical to the one defined in the section Document Risk 4.2.3, resulting in the same desireable properties mentioned above.

$$\texttt{edge\_strength}(d_i, d_j) \;=\; 1 - \prod_{e \in via} (1 - s_e(d_i, d_j))$$

Finally, we prune the graph by removing egdes with `STRENGTH_THRESHOLD` below a user-specified threshold (e.g. 0.3). This reduces computational costs and prevents excessive anonymization of entities during later steps.

Chain (Path) Risk

Re-identification often requires gathering information across multiple documents. To quantify this risk, we extract all *document chains* up to length k (simple paths or multi-document subgraphs) and calculate their respective chain risk. Each *document chain* consists of documents $d_1, \ldots, d_k$ and the edges connecting them. All edges together create a set of entities we refer to as *active entities*. During the anonymization step this set will be partially masked, leaving a reduced set of *active entities*. All following calculations are based on this (potentielly masked) set.

To calculate the Chain Risk $R_{\texttt{chain}}$ of length k, we compute a *hop risk* for each edge/ hop h within the chain H and accumulate it using the complement-of-product formulation. The *hop_risk* of one hop h between two documents d_{i_h} and d_{j_h} is defined as the `edge_strength` scaled by the modified average of both document risks. The scaling captures how much the documents' internal risk amplifies the hop risk, allowing a low average document risk to reduce the *hop risk* by up to 1/2. The exact formula goes:

$$\texttt{hop\_risk}(h) \;=\; \texttt{edge\_strength} * \frac{1 + \frac{R(d_{i_h}) + R(d_{j_h})}{2}}{2}$$

$$R_{\texttt{chain}}(H) \;=\; 1 - \prod_{h \in H} (1 - \texttt{hop\_risk}(h))$$

After obtaining all scores, each chain risk is evaluated and then categorized into a HIGH / MEDIUM / LOW category using `RISK_THRESHOLDS` which define the lowerbound

for each category. Currently, this category serves as a way to validate the pipelines intermediate output. A possible setting is:

$$\texttt{RISK\_THRESHOLDS} = \{\texttt{"HIGH"} : 0.75, \texttt{"MEDIUM"} : ”0.5\}$$

4.2.4 Selective anonymization

The final step of the pipeline performs selective anonymization of entities. Instead of a blanket redaction of all high-risk entities, this step aims to replace the smallest set of entities required to push both single and cross-document (chain) risk below configurable targets. This is realized using a two-stage redaction process: a minimal *document level* redaction pass that removes the largest and most direct risks, followed by a *chain-level* redaction pass, that targets residual chain risks. Throughout, a corpus wide, continuously updated *replacement dictionary* records which entities require masking, ensuring that: (a) already masked entities are excluded from subsequent risk calculations and (b) replacements for entities remain consistent across the entire corpus.

Global entity contribution To prioritize candidates we compute a global importance for each entity e:

$$\text{imp}(e) \;=\; \max_{d \in \text{docs}(e)} \left(\texttt{relevance}(e, d)\right) \cdot u(e) \cdot w_{type(e)}$$

where $\texttt{relevance}(e, d)$ is the per-document utility for re-identification, $u(e)$ the corpus-wide uniqueness (Section 4.2.3), and $w_{type(e)}$ the type-specific risk weight (Table 4.1). This score is primarily used to rank entities during greedy selection algorithms.

Document-level redaction

We begin with redacting entities on a per document basis. For each document d we iteratively select the highest-imp unmasked entity, calculate its pseudonym, update the *replacement dictionary* and recalculate the document risk on the reduced set of entities. This greedy algorithm continues until the risk score is pushed below a threshold θ_{doc} (e.g., 0.95) or no further entities remain.

Rationale. This stage removes the most dominant per-document threats (typically direct identifiers). A greedy selection strategy is appropriate here, as it is fast and focusses only on entities with high imp scores. More nuanced selection at this stage risks protecting a single high-risk entity by masking multiple low-risk ones, going against our objective of preventing a single document from allowing re-identification.

Chain-level redaction

After completing the document-level redaction, we recompute chain risks based on the updated *replacement dictionary* and finalizes chain categories (HIGH / MEDIUM / LOW). Acting on previously calculated categories would impose strong reductions on chains that were already neutralized by the first stage. For chains which remain above policy targets we continue selecting entities to blur with the following stopping rule:

$$\texttt{stop when} \quad R^{new}_{chain} \leq \theta_{chain} \quad \texttt{and} \quad R^{new}_{chain} \leq \rho_{risk\_level}\, R^{pre}_{chain},$$

θ_{chain} acts as an absolute ceiling (e.g. 0.60) and ρ_{risk_level} as a category-specific relative reduction, e.g.

$$\rho_{risk\_level} = \{\texttt{HIGH} : 0.70,\ \texttt{MEDIUM} : 0.90\},$$

This hybrid approach prevents any chain from being above a certain risk using the absolute ceiling, while the relative rule enforces a meaningful reduction of chain risk onto all chains, further ensuring privacy preservation.

Selection strategies. At chain level we consider the following selection strategies:

- **Greedy.** Iteratively mask the current highest-impact entity ($\mathrm{imp}(e)$), recompute affected document and chain risks and repeat until the stopping conditions are met. This is fast and scales to large corpora.
- **Knapsack–style.** Reformulate finding the smallest set of entities to achieve the required risk reduction as a *0/1 knapsack*-like problem. We calculate the required target reduction, the impact of blurring each entity and optimize for the minimum required set using dynamic programming. This approach yields better minimality guarantees and therefore we expect higher utility, but is computationally expensive for higher candidate counts.

Pseudonym generation and replacement

Using the finalized *replacement dictionary* we perform the redaction. Depending on the domain and downstream task, one might choose different anonymization strategies. Our pipeline supports

1. **Complete redaction** (redact using placeholder `[REDACTED]`): Ensures high safety, but removes type information, potentially degrading retrieval and LLM reasoning.
2. **Value redaction** (redact using type label, e.g., "Jane Doe"$\rightarrow$`[NAME]`): Preserves approximate semantics of an entity without retaining linkable identity.

3. **Pseudonymization** (redact using pseudonyms, e.g., `[NAME_hash("JANE DOE")]`): Preserves references across mentions, but also preserves cross–document linkability.

4. **LLM rewriting**. Instruct a model with rewriting while omitting selected entities. Difficult to audit as document structure might change and hallucinations might appear.

We adopt **value redaction** as the default as (i) it maintains the semantic role of an entity for downstream tasks while breaking up cross-document linkability that pseudonymization would retain. **Pseudonymization** could be used in scenarios, where consistency of references to entities are important, even when the value itself is not available and don't increase the privacy risk significantly. An example would be internal process IDs with low external linkage potential. When pseudonyms are used, they are deterministically generated by appending their hashed entity id to their type, allowing corpus wide consistent referencing while keeping the semantic of an entity:

$$\texttt{pseudonym}(e) \;=\; e.type|hash(e.id)$$

For strategy 1 - 3 the we perform whole-word, case-insensitive matching of extracted original values (Section 4.1) to documents containing the entities.

4.2.5 Hyperparameters

In this section we will provide reasoning and evidence for hyperparameters. A small train-dataset is generated using a python script we introduce in the chapter Evaluation (Section 5.1.1). This train-dataset includes 5 sets of documents (also called "clusters"), with each set containing 3-6 documents for a total of 29. Each cluster thematically focusses on one topic and "hides" one person, which will become relevant during the evaluation.

We plan a multi-stage tuning process, evaluating the performance of the model using three checkpoints in the pipeline. At each checkpoint, the models output is compared to an expected answer using a similarity measure. To obtain this expected answer the dataset is manually annotated.

Tuning: Context Entity Filtering

The goal of this section is to find the appropriate settings for filtering the context entities before the second extraction stage. The context passed to the second extraction step must contain all relevant entities required to uncover cross-document links, while being compact enough to avoid prompt bloat and a potential degradation in performance.

As a baseline, we manually extract the entities from each document that the Entity Extractor might deem relevant for re-identification or linking. This baseline is then compared against the output of the pipeline after the Context-Aware Entity Extraction (see Section 4.2.2).

We evaluate the quality of entity filtering using *ROUGE-1 F1* on a per-document basis and average the results to obtain the final score for a given filter setting. *ROUGE-1 F1* is chosen because it provides a balanced trade-off between precision and recall, ensuring that missing relevant entities (low recall) and too many irrelevant ones (low precision) are penalized. This suits our task, as both types of errors degrade linking quality by either failing to capture links between documents or introducing noise.

The configurable parameters in our implementation are:

- the strength of the heuristic filter `ENTITY_FILTER_STRENGTH` (continuous between 0 and 1),
- the use of type-based filtering (enabled/disabled).

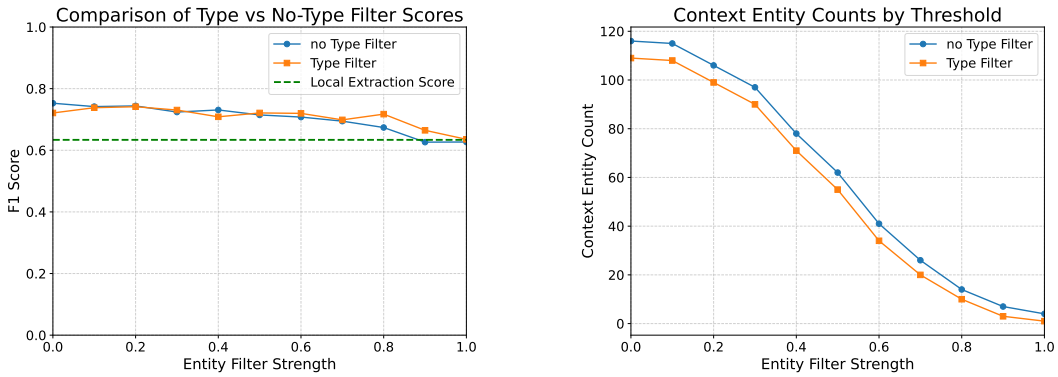
It should be noted that, due to the stochastic nature of LLMs, results can vary between runs. To combat this issue, we run each threshold configuration three times and average their scores. This stabilizes some fluctuations, but still isn't enough to prevent them entirely. Additionally, the current train-dataset is too small to cause potential degradation during the second stage, resulting in high scores for low or no filtering thresholds. Therefore we only focus on finding the highest threshold that contains most of the relevant entities.

Graph (a) shows that the `ENTITY_FILTER_STRENGTH` is inversely correlated with the extraction quality of the second extraction stage. Both lines (orange & blue) stay consistent and start to drop off around 0.4. Regardless of the type-filter setting the extraction performance stays similar across all thresholds, indicating that the assumption made earlier (Section 4.2.2) was correct. As a result we select `ENTITY_FILTER_STRENGTH` < 0.4 and activate filtering by entity type to reduce the number of context entites.

Justification of the two-stage extraction can be found when comparing the green horizontal line (results after first extraction stage) with the results from the second extraction stage. The horizontal line consistently lies below the stage two results, only coming close when the filter removes almost all context-entities.

Tuning: Graph Construction and Chain Risk

We attempt to find the optimal parameters to extract all required *document chains* which might be useful for reidentification. First, we measure the quality by comparing the number of connections between documents from the same cluster (intra-cluster



(a) *ROUGE-1 F1* across thresholds.

(b) Entity counts across thresholds.

Figure 4.1: Comparison of type-based and non-type-based filtering. (a) shows performance using *ROUGE-1 F1* while (b) illustrates the corresponding entity counts.

connection) and documents from different clusters (inter-cluster connection). We aim to minimize inter-cluster connections while preserving most of the relevant intra-cluster connections.

The configurable parameter in this case is the `EDGE_STRENGTH_THRESHOLD`, which removes edges with too little strength. These are often inter-cluster connections, as documents about different topics typically share less entities. Removing these weak links prevents over-anonymization during the anonymization of entities. For example: three documents *doc1*, *doc2*, *doc3* form a chain, *doc1*, *doc2* are strongly linked with a weak link to *doc3*. If an entity is blurred in *doc1*, *doc2* it will also be blurredin doc3, despite *doc3* not being a risk, therefore potentially harming utility without improving privacy.

For this train-dataset, a reasonable interval for the `EDGE_STRENGTH_THRESHOLD` is $[0.2, 0.5]$. Within this interval a healthy number of intra-cluster connections remain, while inter-cluster connections drop steadily.

Using this interval as a starting point, we perform a grid-search including the surrounding intervals to find a final setting for the `EDGE_STRENGTH_THRESHOLD` and the optimal value for the `RISK_THRESHOLD`. For simplicity we focus on the threshold defined for "MEDIUM", as all *document chains* above this threshold will be added as risks. We aim to find out whether the pipeline was able to identify and keep all relevant connections between documents up until this point. Our similarity measure is the F1-Score between all *document chains* marked as dangerous by the pipeline and the hand-labeled *document chains*.

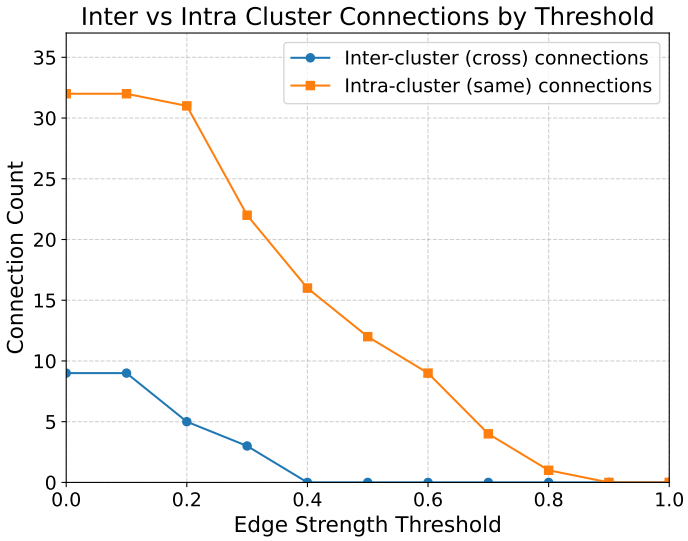


Figure 4.2: Graph connections across `EDGE_STRENGTH_THRESHOLD`

Out of all different combinations, the setting: `EDGE_STRENGTH_THRESHOLD` $= 0.5$ performs the best, with a maximum F1-Score of 0.776. Unexpectedly, this F1-Score is constant in the interval `RISK_THRESHOLD` $\in [0.0, 0.5]$, suggesting that for this example the pipeline didn't extract lower-risk chains which could be filtered out by setting `RISK_THRESHOLD` below 0.5. Afterwards it begins to fall, indicating that relevant chains are being filtered out. We use this setting as our final setting for the pipeline. The reported Recall and Precision can be found in the appendix ()

Tuning: Anonymization Thresholds

The following section focusses on the the anonymization thresholds θ_{doc} and θ_{chain}. We skip an extensive search for optimal ρ_{risk_level} as this relative reduction is only applied as an additional measure to increase privacy. Again, we have defined a set of entities which we deem, when redacted, provide sufficient privacy protection.

4.3 RQ1: Can the proposed pipeline model and cross-document linkage sufficiently

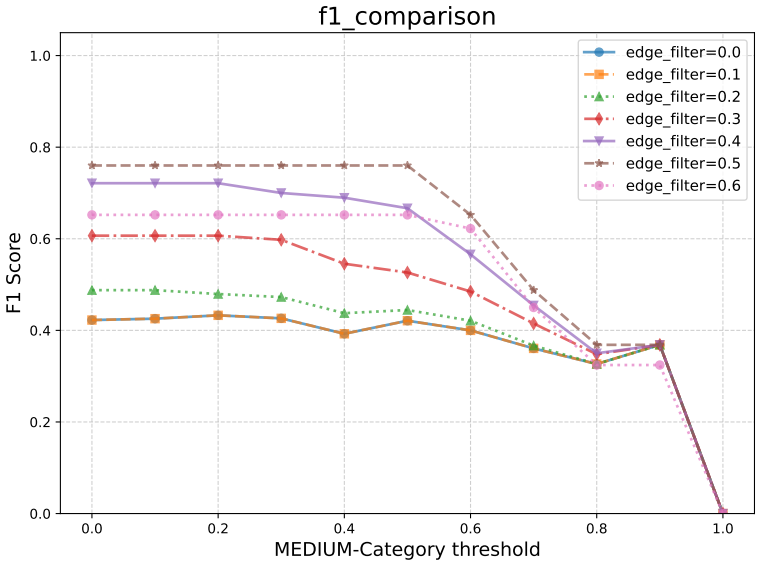


Figure 4.3: F1 Score for different combinations of `RISK_THRESHOLD` and `EDGE_STRENGTH_THRESHOLD`

5 Evaluation

5.1 Experimental Setup

5.1.1 Data Generation

5.1.2

5.2 RQ2: Does selective pseudonymization reduce privacy leakage on both the document and cross-document linkage level

5.3 RQ3: Does selective pseudonymization preserve higher utility compared to similar approaches

6 Discussion

6.1 Limitations

7 Conclusion

ff

Abbreviations

TUM Technical University of Munich

List of Figures

List of Tables

Bibliography

[Lam94] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.