# Offline Multimodal AI Redaction of Sensitive Data in Audio and Software Artifacts

**Etienne Kaiser**[1] **and Nico Heiniger**[2]

[a]Prof. Dr. Daniel Patrick Politze; [b]INS Institute for Network and Security ; [c]Eastern Switzerland University of Applied Sciences (OST)

This paper was drafted on December 19, 2025

**This research investigates the feasibility of an offline, open-source system for automatic detection and redaction of sensitive data in both audio transcriptions and software artifacts. Cloud-based services for Speech-to-Text (STT) transcription and data redaction raise privacy concerns for organizations handling personally identifiable information (PII) or authentication secrets. The prototype developed combines Faster Whisper for speech recognition with Microsoft Presidio and custom pattern recognizers for entity recognition and redaction. Two experiments evaluate system performance: (1) comparing PII redaction accuracy between text-only and STT-transcribed inputs, and (2) assessing secret detection in log and code files. Results demonstrate 97% recall for text-based PII redaction and 99% for audio-transcribed content, with the STT pipeline introducing a slight masking effect through transcription variations. Secret redaction achieves 90% recall, with challenges in detecting high-entropy tokens like API keys. The findings confirm the viability of offline redaction pipelines while identifying domain-specific limitations that allow further research into fine-tuned machine-learned detection models.**

Redaction | PII | Secrets | Speech-to-Text | Entity Recognition | Privacy

## List of Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| ASR | Automatic Speech Recognition |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| LLM | Large Language Model |
| NER | Named Entity Recognition |
| NLP | Natural Language Processing |
| PII | Personally Identifiable Information |
| RQ | Research Question |
| SDK | Software Development Kit |
| STT | Speech-to-Text |
| VAD | Voice Activity Detection |

## 1 Introduction

In modern applications such as AI assistants and chatbots, spoken communication plays an essential role. Voice interactions are frequently stored as raw audio recordings during processing, creating opportunities for unintentional disclosure of sensitive information. Such data may include personal details and confidential business information, additionally, log files may contain authentication secrets.

Cloud-based Speech-to-Text (STT) services have achieved remarkable accuracy in recent years, providing high-quality transcription across numerous languages and domains. However, these systems typically require data upload to remote servers for processing. For organizations operating under strict compliance regulations or frequently handling Personally Identifiable Information (PII), this procedure raises serious concerns about confidentiality, legal compliance and data sovereignty. Even when cloud providers maintain high privacy standards, the mere transfer of data outside controlled infrastructure may violate internal policies or regulatory requirements.

Consequently, many organizations seek local, self-hosted solutions that prevent data exposure to third-party providers while enabling sensitive information detection and redaction entirely on-premises. Such solutions would allow data containing sensitive information to be processed and anonymized before leaving secure environments, making them particularly attractive for highly regulated industries including cybersecurity, finance and healthcare.

### 1.1 Problem Statement

Existing open-source speech recognition models, such as Whisper [1] and Vosk [2], lack integrated mechanisms for automatically detecting and redacting PII real-time, as well as during post-processing. This creates an opportunity, while modern transcription technologies are powerful, privacy-focused enterprises cannot fully utilize them without risking exposure of sensitive data. Many organizations aim to adopt AI-powered solutions to enhance workflows but are facing obstacles by this conflict. Consequently, innovation is often postponed, limited, or - especially in smaller companies - completely abandoned.

### 1.2 Objective

This study addresses this discrepancy by developing a local prototype using only open-source components, capable of both transcription and automatic redaction of PII from spoken audio streams as well as from plain text input such as log files and code snippets. Redaction in this context refers to the anonymization of personally identifiable or otherwise confidential data like secrets, ensuring sensitive content is filtered before further processing by third-party applications. A working prototype would demonstrate the possibility of a solution allowing companies to integrate automated audio-processing pipelines internally. Enabling users to filter data before interacting with chatbots or similar services. This reduces information leakage risk and supports data protection as early as possible in internal data-processing workflows. Derived from the main research question, *How reliably can sensitive data in source code and log files be detected and automatically masked?*, two specific objectives (Figure 1) are examined.

**RQ1:** How does an offline STT transcription affect textual redaction of PII?

**RQ2:** How effectively can an offline system redact secrets in software artifacts?

**Fig. 1.** Derived research questions from core objective.

### 1.3 Scope and Limitations

The primary challenge lies in the absence of products that automatically and reliably redact sensitive data in audio, transcribed text, code or log files. Audio files must first be processed through STT models to accurately capture and subsequently redact content. Due to latency constraints and CPU system requirements, real-time redaction is excluded from this work, instead, audio recording with subsequent manual verification of transcription and sequential redaction execution serves as the primary use case. An external interface, specifically a LLM powered by Mistral AI [3], provides feedback of how redacted text might be processed by online services.

Based on the defined research questions, the term offline in this context means the core transcription and redaction functionality operates without network access, while open-source indicates all components are freely available with accessible source code.

Sensitive data in this context is defined as PII and Secrets relevant to authentication, cybersecurity and software artifacts. Other sensitive data categories, such as Proteced Health Information (PHI), Business & Confidential Data, Financial Data or Biometric Data [4] may partially appear but fall outside the current scope.

Given the variety of audio and text file formats, support is limited to specific types detailed in the System Requirements chapter. Another limitation are computational resources with the target being CPU execution rather than GPU usage, covering a broader range of potential prototype deployment scenarios.

For transcription, all 99+ languages supported by Faster Whisper [5] (based on OpenAI's Whisper [1]) are available. For redaction, however, the prototype is limited to English and German, extension to additional languages remains straightforward.

### 1.4 Paper Structure

This paper is structured into five main chapters, each building upon the introduction and guiding readers from problem motivation through implementation and evaluation to conclusions.

**Chapter 2** presents current technologies and research, including speech recognition systems, PII detection methods, anonymization strategies, as well as commercial and open-source tools. It positions the work within the broader technical landscape and identifies the gaps addressed by the proposed prototype.

**Chapter 3** describes methodology, including system requirements supporting investigation of the research questions. An overview of design decisions and implementations covers audio processing with STT modules and redaction capabilities.

**Chapter 4** analyzes the prototype through experiments examining redaction accuracy. The chapter compares text-based redaction with STT-based redaction and explores how variable changes affect PII detection performance.

**Chapter 5** presents conclusions, summarizing findings from experiments, addressing research questions and providing recommendations for secure redaction of sensitive data using machine learning.

## 2 Related Work and Gap Analysis

This section reviews research relevant to this study with highlighting current capabilities, identified existing limitations and reveals gaps that motivate the development of an offline, open-source redaction prototype.

*Transforming Redaction: How AI is Revolutionizing Data Protection* Peng et al. [6] present an experiment comparing manual redaction, classical machine learning methods and AI-assisted redaction tools. The AI approach achieves higher accuracy and faster completion times than manual methods, while classical machine learning still requires manual intervention for certain sensitive data types. The authors point out AI-assisted redaction can reduce human error and improve regulatory compliance, though further work is needed to generalize across document types and contexts. These results motivate the need for automated redaction while highlighting limitations of current solutions.

Pelikan et al. [7] mention in their Apple Machine Learning Research paper that establishes *"the first benchmark for Federated Learning (FL) with DP in end-to-end ASR."* The authors emphasize that *"the application of automatic speech recognition (ASR) remains largely unexplored"* in privacy preserving contexts and note that *"no existing work establishes a competitive, practical recipe for FL with DP in the context of ASR."* This is confirming the critical absence of offline, privacy-preserving STT pipelines (Figure 2).

Soveizi et al. [8] highlight that security remains a critical barrier to cloud adoption for workflows dealing with sensitive data, emphasizing the need for on-premise alternatives where data cannot leave local infrastructure.

Nautsch et al. [9] establish requirements for privacy preservation in speech data and notes that *"very few solutions have been proposed to protect privacy in the case of speech signals"* despite regulatory mandates, supporting the gap in privacy preserving STT systems.

Bell et al. [10] introduce the task of audio de-identification in medical records, proposing a pipeline combining Automatic Speech Recognition with Named Entity Recognition to locate and mask PII in spoken content. The authors highlight challenges such as alignment between transcripts and audio segments (Figure 2), with experiments showing ASR errors significantly reduce detection accuracy, emphasizing the need for robust STT models.

Mishra et al. [11] mention that *"current methods for PII detection and anonymization approaches are limited by inflexible criteria, significant pre-processing, high computational costs"* and emphasizes the need for *"a scalable, integrated NLP-ML strategy"* that is currently lacking.

Berzi et al. [12] provide an overview of methods for removing PII from clinical free-text documents. Their analysis shows a clear shift from early rule-based systems to advanced deep-learning approaches, with hybrid models combining rules and neural methods appearing in recent years. Although many systems report high $F_1$-scores on benchmark datasets, results often do not generalize well to new domains. Key challenges are limited availability of correctly annotated data, poor cross-domain robustness and the need for realistic evaluation metrics reflecting current privacy risks.

Rahman et al. [13] demonstrate significant limitations of current secret detection tools, showing they miss secrets embedded in logs and URL-encoded formats. Notes that *"scan-*

*ners requiring clean boundaries around the token will miss this even if the secret is valid."* supporting the gap in comprehensive secret detection.

Meli et al. [14] conducted a large-scale study of API key and secret leakage in GitHub repositories, documenting extensive exposure of authentication tokens and identifying leak patterns across commits, configuration files and build logs. They demonstrate that secrets are harvested by attackers shortly after public appearance. This work highlights the importance of automated secret detection and validates the need for redaction in software artifacts (Figure 3).

Templ et al. [15] note that comprehensive approaches for real-world scenarios remain critically absent, leaving practitioners without clear guidance on combining techniques to address *"multi-faceted privacy requirements"*.

While prior work has explored automated redaction in text documents, audio de-identification using ASR, and the detection of leaked secrets in software artifacts, none of the reviewed approaches integrate these steps into a unified, offline redaction pipeline covering speech transcription, PII identification and secret detection (Figure 4).

## 2.1 Commercial PII Redaction Solutions

This section provides an overview of existing tools and products currently available. The focus is purely descriptive, detailed gap analysis follows at the end of this chapter.

### 2.1.1 Text-based systems

Textual PII redaction refers to automated detection and masking of sensitive information identifying individuals in written text. Redaction systems typically rely on combinations of rule-based techniques (e.g. regular expressions, short regex [16]), Named Entity Recognition (NER) and machine-learning-based classifiers.

Microsoft Presidio SDK [17] is a modular framework for detecting and anonymizing PII in text. It combines rule-based recognizers, regular expressions and NER models while allowing custom detectors and masking. Presidio is a flexible open-source solution available for local PII redaction.

AWS Macie [18] is a fully managed AWS service automatically scanning and classifying data stored in Amazon S3, identifying sensitive data using machine learning and pattern matching. It focuses on large-scale data and operates exclusively within AWS cloud environments.

Google DLP (Data Loss Prevention) [19] provides an API for PII detection supporting various data sources and formats, offering predefined detectors and supports masking, tokenization and transformations. All processing is done on Google Cloud [20].

spaCy [21] is an NLP library offering high-performance NER models adaptable for PII detection. While it lacks dedicated PII components, many open-source or commercial pipelines build upon spaCy due to its speed and performance.

NVIDIA GLiNER [22] (part of NVIDIA NeMo [23]) is a high-performance NER system optimized for GPU inference, according to Huggingface [22] also CPU support, it allows customizable NER pipelines. It can be fine-tuned on domain-specific datasets and deployed fully offline on CPU or NVIDIA GPU hardware (cuda [24]).

| Product | Offline | Open-Source |
|---|---|---|
| Microsoft Presidio SDK | Yes | Yes |
| AWS Macie | No | No |
| Google DLP | No | No |
| spaCy | Yes | Yes |
| NVIDIA GLiNER | Yes | Partial |

**Table 1. Comparison of text-based PII detection products.**

Table 1 highlights that cloud-based PII detection solutions often lack transparency, while offline-capable alternatives are more commonly open-source.

### 2.1.2 STT-based systems

STT systems convert spoken audio into written text using deep learning techniques such as encoder-decoder architectures or transformer-based models to achieve high transcription accuracy. STT is an essential component for audio-processing pipelines aiming to analyze or redact spoken content. Solutions range from cloud-based commercial or open-source products to fully offline models (Table 2).

AWS Transcribe Call Analytics [25], Google Cloud Speech-to-Text [26] and Microsoft Azure Speech Service [27] are cloud-based speech processing service designed for conversational analytics, providing automated STT transcription combined with call categorization, sentiment analysis and sensitive data redaction.

Deepgram [28] offers a STT platform based on end-to-end deep neural networks optimized for high accuracy and real-time processing. Pretrained and domain-specific models are accessible mainly via cloud APIs, with on-premises deployment available only for enterprise subscriptions.

Speechmatics [29] provides open-source SDKs for interacting with its API, while the speech recognition engine itself remains a closed-source commercial product that also offers offline capabilities.

OpenAI Whisper is an open-source transformer-based STT model providing multilingual transcription performance, noise robustness and high accuracy. Whisper runs fully offline on GPU or CPU, making it suitable for regulated applications, though larger models require substantial computational resources.

Vosk is a lightweight and offline open-source STT project designed for embedded systems. Supporting many languages Python, Java, C# and other environments. Vosk is frequently used in resource-constrained settings due to low computational requirements.

| Product | Offline | Open-Source |
|---|---|---|
| AWS Transcribe Call Analytics | No | No |
| Google Cloud Speech-to-Text | No | No |
| Microsoft Azure Speech Service | No | No |
| Deepgram | Partial | No |
| Speechmatics | Yes | No |
| Whisper (OpenAI) | Yes | Yes |
| Vosk | Yes | Yes |

**Table 2. Comparison of STT-based PII detection products.**

## 2.2 Available Secret Redaction Solutions

Secret redaction focuses on identifying and removing sensitive data such as API keys or private keys and other authentication tokens appearing within software artifacts. Exposed tokens can lead to unauthorized system or data access. Secret redaction tools typically use pattern matching or signature-based methods. This domain has gained significant relevance due to increased secret leakage in public repositories and automated pipelines, as previously shown by Meli et al. [14].

### *2.2.1 Text-based systems*

TruffleHog [30] detects high-entropy strings and known credential patterns in repositories, file systems and text. It supports scanning Git history, including commits and is used for discovering leaked secrets such as API keys and secrets in developer environments.

AWS CodeGuru Secrets Profiler [31] identifies secrets such as hardcoded credentials, tokens and database passwords within code. It integrates with AWS developer tools, performs static analysis and provides recommendations.

detect-secrets (Yelp) [32] is a plug-in for secret detection preventing new secrets from entering version control systems. It uses heuristics, regex patterns and baseline files to minimize false positives and integrates with Git pre-commit hooks.

| Product | Offline | Open-Source |
|---|---|---|
| TruffleHog | Yes | Yes |
| AWS CodeGuru Secrets Profiler | No | No |
| detect-secrets (Yelp) | Yes | Yes |

**Table 3. Comparison of text-based secret detection products.**

As shown in Table 3, fewer existing solutions for secret detection and redaction exist, with only one reknown cloud-based and closed source service.

## 2.3 Identified Gaps

The review of the market and related work uncovered several gaps within existing technologies, motivating the research questions.

Lack of an offline and open-source STT redaction pipeline.

**Fig. 2.** First identified research gap.

Most existing solutions rely on cloud-based STT or redaction APIs and are optimized for large-scale data processing. Although several open-source STT models exist, none provide a fully integrated, offline redaction pipeline (Figure 2). Current systems either depend on external services, are not open-source, or lack PII detection. This is a critical limitation for privacy environments where data must remain on local hardware and will be examined in RQ1.

Missing offline secret redaction for software artifacts.

**Fig. 3.** Second identified research gap.

Existing secret detection tools are mainly built for version control systems and code repositories. They do not address secrets appearing in logs, documents, transcripts or text copied into chatbots common real-world scenarios (Figure 3). Additionally, the most capable tools are cloud-based, making them unsuitable for offline use cases, motivating RQ2.

No integrated offline pipeline for sequential STT, PII and secret redaction.

**Fig. 4.** Third identified research gap.

Even standalone products exist for transcription, PII detection and secret scanning, there is no unified offline pipeline that combines all three capabilities (Figure 4). Organizations currently must build custom solutions, which introduces inconsistencies, privacy risks and financial overhead.

## 2.4 Research Motivation

In summary, current solutions do not meet requirements of an offline, open-source STT and redaction pipeline handling both personal and security sensitive information. These gaps highlight the need for investigation covered by the formulated research questions.

# 3 Methodology

This chapter describes the prototype developed and the following sections describe system requirements, architecture, implementation details and design decisions based on experienced challenges.

## 3.1 System Requirements

The prototype must support uploading both audio and text files for transcription, subsequent editing and redaction before transmission to external sources. Various common-known file formats are supported and internally converted (Table 4).

| Input Type | Supported Formats |
|---|---|
| Audio Files | MP3, WAV, M4A, FLAC, OGG, OPUS, WebM |
| Textual Files | TXT, MD, LOG, PY, JSON, CSV |

**Table 4. Supported file formats by input type that can be loaded.**

Key system requirements derived from the objective to investigate include REST API endpoints, PostgreSQL [33] database for session persistence, web interface, Docker containerization for deployment, CPU only execution without dependence on GPU and support for English and German language redaction.

## 3.2 Architecture Overview

The prototype consists of two main modules for redacting audio and text input. Text processing feeds raw input directly into the text redacting module, generating redacted text output. Audio input requires transformation to text as a first step through the Speech-to-Text module, which transcribes audio and forwards output to the same text redacting module. Both input types produce the same output form, redacted

text, suitable for safe further processing, including transmission to chatbots or other external services.

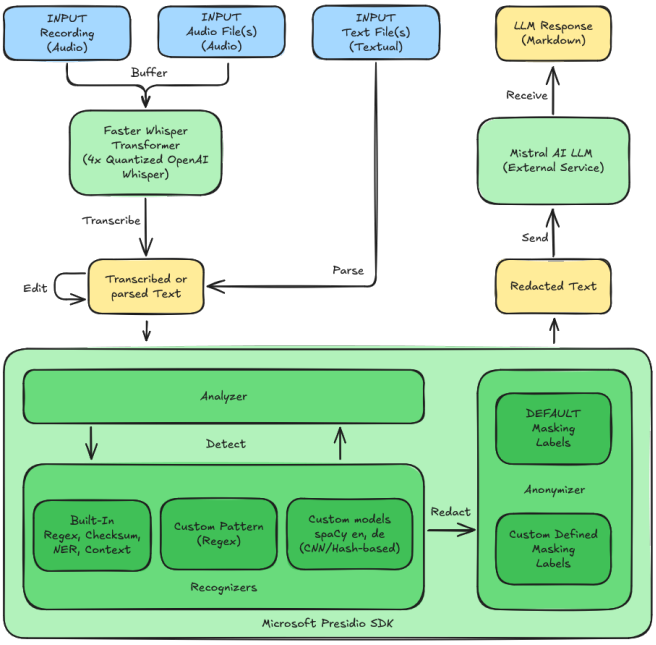


**Fig. 5.** Architecture and workflow of processing transcription, detection and redaction.

Figure 5 shows the explained workflow as well as an overview of the core technologies (Table 5) that influenced the architecture.

| Technology | Task |
|---|---|
| Faster Whisper | Multilingual STT transcription on CPU |
| Microsoft Presidio SDK | PII detection and anonymization |
| spaCy NER Models | Foundational entity detection using NER |
| Custom Regex Engine | Extended pattern recognition |

**Table 5. Responsibility of used technologies in developed prototype.**

## 3.3 Implementation Details

The following sections demonstrate specific implementation approaches for each major component, focusing on technical choices rather than code specifics.

### *3.3.1 Audio Processing and STT module*

Audio processing utilizes Faster Whisper, a reimplementation of OpenAI's Whisper model. The medium model size balances accuracy and resource consumption running on CPU. Audio files using preprocessing through FFmpeg [34], converting various input formats to 16kHz (downsampling) mono as model input.

The transcription pipeline includes Voice Activity Detection (VAD) filtering to improve accuracy by focusing on speech segments. Language detection is done automatically, though manual override in setting would be available. The model supports 99+ languages for transcription, with detected language information passed to the redaction module (currently either English or German) for appropriate model selection. Transcription results are returned as concatenated text segments when uploading multiple files or pausing during recording, preserving natural speech flow.

### *3.3.2 Redaction module*

The redaction module relies on Microsoft Presidio SDK which combines multiple detection approaches: built-in recognizers for common PII types, integration with spaCy NER models for entity detection and extensible pattern recognizers for custom data types. The implementation loads spaCy's *en_core_web_lg* [35] (English) and *de_core_news_lg* [36] (German) models, providing Named Entity Recognition capabilities for persons, locations, organizations and dates. These models serve as the foundation for Presidio's entity detection, with automatic language selection based on text content. Anonymization replaces detected entities with labeled placeholders (e.g. [PERSON]).

### *3.3.3 Backend and Frontend*

The backend is mostly written in Python [37], with FastAPI [38] exposing endpoints for audio and text upload, transcription, redaction, LLM queries and session management. SQLAlchemy [39] abstracts database access with PostgreSQL storing session data such as original text, redacted text and LLM outputs.

For simplicity, Figure 6 provides a high-level visual impression of the single-page application (SPA). The figure is intended for illustrative purposes only, focusing on the general layout and appearance of the prototype rather than detailed functionality.

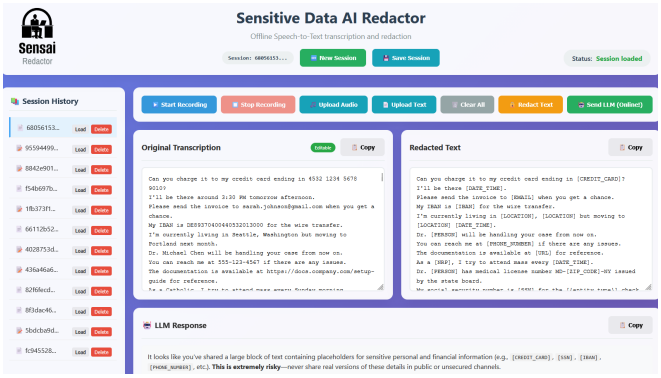


**Fig. 6.** Frontend of the prototype with comprehensive User Interface and Session Management

Docker Compose orchestrates the multi container deployment with a container each for PostgreSQL database, FastAPI backend with model loading and NGINX [40] frontend services.

## 3.4 Design Decisions and Challenges

Several design decisions shaped the prototype implementation, each addressing specific technical or practical challenges.

**Sequential Processing Workflow:** The design intentionally separates transcription and redaction into distinct manually triggered steps. This allows users to review and correct transcription errors before redaction under assumption that STT models may introduce errors affecting detection. Additionally, the sequential approach also enables text only use cases without audio processing upfront.

**spaCy NER Models over Transformer Models:** While transformer-based models like GLiNER from NVIDIA offer potentially higher accuracy, spaCy's smaller models (*en_core_web_lg* at ~500MB) provide sufficient entity detection for standard PII types while maintaining reasonable memory consumption. Testing showed transformer models provided no measurable improvement for standard PII labels, as these entities are well represented in classical NER training data. Additionally, secrets require custom pattern recognition regardless of NER model choice, making the added complexity and resource requirements of larger models unjustified.

**Faster Whisper over OpenAI Whisper:** Initial development used OpenAI's original Whisper implementation, but Faster Whisper provides approximately 4x memory reduction through CTranslate2 [41] optimization and int8 quantization. This enables CPU only deployment scenarios without sacrificing transcription accuracy but at the same time broadening potential deployment environments.

**Presidio SDK over PIIranha [42]:** Evaluation of PIIranha, an open-source PII detection model, revealed insufficient secret detection capabilities. Since custom regex patterns were required anyway, Presidio's extensible architecture with built-in pattern recognizer support proved more fitting. Memory consumption considerations further favored Presidio's modular approach, as PIIranha as transformer initially ate up a large chunk of total used ressources.

**Custom Regex for Secrets:** Standard NER models lack training for cybersecurity related secrets appearing in code and logs. Rather than fine-tuning models for this domain, pattern-based recognition was implemented for known secret formats. This approach provides deterministic detection for well-known patterns while acknowledging limitations for high-entropy tokens (like API keys) without predictive formats.

**CPU over GPU:** Explicitly prefering CPU execution ensures development in bussiness context on laptops to lightweight servers without GPU access. While GPU acceleration would improve processing speed, the latency tolerant batch processing model makes CPU execution acceptable for the intended use cases.

## 4 Experimental Validation

This chapter presents experimental validation of the prototype system, designed to address the research questions through empirical measurement. The methodology, datasets, metrics and results are detailed in the coming sections.

PII redaction accuracy for audio transcribed text will be lower than for direct text input, as multimodal inaccuracy in transcription prevents exact reproduction of original text.

**Fig. 7.** Hypothesis for Experiment 1.

The developed prototype will significantly improve detection and redaction of secrets in code and log files compared to standard PII solutions but a gap will still remain.

**Fig. 8.** Hypothesis for Experiment 2.

Two initial hypotheses guide the experimental design and will be validated (Figure 7 and Figure 8).

### 4.1 Experimental Design

Two independent experiments evaluate system performance by comparing original unredacted files against redacted outputs to determine how many sensitive data items were successfully redacted. The prototype processes uploaded files and generates redacted outputs for comparison against ground truth. Partially (longest common subsequence might be an option in future experiments) or fully redacted sensitive data items are counted as redacted, only completely unmasked items count as failures.

#### *4.1.1 Experiment 1: Textual PII redaction vs. STT transcribed PII redaction*

The first experiment addresses RQ1, investigating differences between Speech-to-Text transcribed text redaction afterwards versus text only redaction. Text-based redaction serves as baseline for comparison against audio transcribed redaction, enabling measurement without noise of transcription effects on redaction performance.

#### *4.1.2 Experiment 2: Secret redaction in software artifacts*

The second experiment targets RQ2, examining how effectively the system detects and redacts secrets in log files and code. This experiment evaluates performance against software specific secret types typically not covered by standard PII detection systems.

### 4.2 Dataset

A dedicated dataset was created for each experiment. The first dataset contains 108 labeled test sentences with sensitive data that can be both spoken and written, enabling realistic comparison scenarios.

The second dataset contains five log file types (Table 6), containing 150 instances of sensitive data, namely secrets, typically found in logs or code snippets that usually appear textually rather than spoken, such as SSH_KEY.

| File Type | Description |
|---|---|
| Python Logging | Standard application logging format |
| JSON API Logs | Structured JSON logging output |
| Shell/Bash Scripts | Unix shell scripts with credentials |
| Configuration Files | YAML, INI, ENV configs |
| SQL/Database Logs | Database connection and query logs |

**Table 6. Test set for benchmarking of Experiment 2.**

#### *4.2.1 Sensitive data type selection*

Experiment 1 focuses on PII data types commonly encountered in human communication, while Experiment 2 addresses secrets specific to software development and system configuration. As shown in Table 7, the data types evaluated in each experiment are distinct and address different categories.

| Experiment 1: PII Types | Experiment 2: Secret Types |
|---|---|
| BANK_ACCOUNT | API_KEY |
| CREDIT_CARD | AWS_ACCESS_KEY |
| DATE_TIME | AWS_SECRET_KEY |
| DRIVER_LICENSE | BEARER_TOKEN |
| EMAIL_ADDRESS | DATABASE_URI |
| IBAN | GITHUB_TOKEN |
| ID_CARD | GOOGLE_API_KEY |
| LOCATION | IPV6 |
| MEDICAL_LICENSE | IP_ADDRESS |
| NRP | JWT_TOKEN |
| PASSPORT | MAC_ADDRESS |
| PERSON | PASSWORD |
| PHONE_NUMBER | PRIVATE_KEY |
| SSN | SENDGRID_API_KEY |
| TAX_ID | SLACK_TOKEN |
| URL | SSH_KEY |
| USERNAME | STRIPE_KEY |
| ZIP_CODE | UUID |

**Table 7. Alphabetically sorted data types for test sets of both experiments.**

#### 4.2.2 Data origin and generation

The experimental data were synthetically generated based on real-world knowledge and experience, with sensitive data types selected to ensure balanced representation.

#### 4.2.3 Data transformation and labeling

Data was manually labeled and for efficiency, individual sentences (PII experiment) and log file snippets (secrets experiment) were aggregated into single files for batch redaction processing. Ground truth files maintain original sensitive values to check against their expected label category.

### 4.3 Evaluation

Data evaluation uses precision and recall metrics computed in Jupyter [43] notebooks. During metric implementation, a critical observation was made: evaluating by matching expected labels to redacted labels would produce misleading results. Labels do not reliably map back to original sensitive data because redaction may assign different label names to the same underlying data type. For instance, a phone number might be labeled as [PHONE_NUMBER] by one recognizer but [CONTACT] by another. Consequently, evaluation is done by actual value comparison: checking whether original sensitive data strings (regardless of label assignment) remain visible in redacted output. This approach verifies whether original sensitive data was actually redacted, independent of label naming conventions. Labels serve only to associate original values with their expected categories for analysis purposes, not as the primary success criterion. This methodology avoids the problematic effects of working with pre-defined Presidio SDK labels enriched by custom label as done with regex, where different recognizers might correctly redact the same data under different names. The fundamental question - *Is the sensitive value still visible?* - serves as the guiding principle throughout this study and provides the most meaningful measure.

#### 4.3.1 Metric Definitions

Based on the value-level evaluation strategy described above, the following confusion matrix terms are defined as follows:

- **True Positive (TP):** A sensitive value that is expected and is successfully redacted in the output.
- **False Negative (FN):** A sensitive value that is expected but remains visible in the redacted output.
- **False Positive (FP):** A value that is not expected to be redacted but is nevertheless masked by the system (over-redaction).
- **True Negative (TN):** A non-sensitive value that is correctly left unredacted.

In practice, true negatives are not explicitly enumerated, as the evaluation focuses on sensitive values and their redaction behavior rather than on the exhaustive amount of non-sensitive data.

#### 4.3.2 Evaluation Metrics

Using these definitions, system performance is measured using precision, recall and the $F_1$-score, which are standard metrics in information retrieval and classification tasks [44].

Recall quantifies the systems ability to identify and redact all sensitive values and is defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad [1]$$

Precision measures how many of the redactions performed by the system are correct and is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad [2]$$

The $F_1$-score is the harmonic mean of precision and recall and provides a single metric balancing over-redaction and missed detections:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad [3]$$

In the context of redaction, recall is particularly critical, as missed sensitive values represent a direct privacy risk. Precision remains important to limit excessive redaction, which may reduce resulting data quality and readability. The $F_1$-score therefore serves as a compact indicator of overall redaction effectiveness.

### 4.4 Experimental Procedure

For Experiment 1, audio files are uploaded to the prototype, transcribed by Faster Whisper and manually triggered for redaction. Redacted text is then compared against ground truth in Jupyter notebooks. For the text only baseline, original text files are directly redacted without transcription. Experiment 2 follows the same procedure without the transcription step, as secrets data is by nature text-based.

### 4.5 Results

Results for Experiment 1 show highest coverage, 99% of PII values were redacted, though label assignments were sometimes incorrect or partially correct as some values received different category labels (Figure 15). Additionally, some non-PII data were marked as sensitive (false positives).

**Text-based PII Redaction:** The system achieving 97% coverage but struggled with usernames (66.7%) and NRP (Nationality, Religion, Political group, 83.3%). All other PII categories achieved 100% detection coverage (Figure 16).

**Audio-transcribed PII Redaction:** Following evaluates redaction results on audio-transcribed input and contrasts them with the textual baseline.

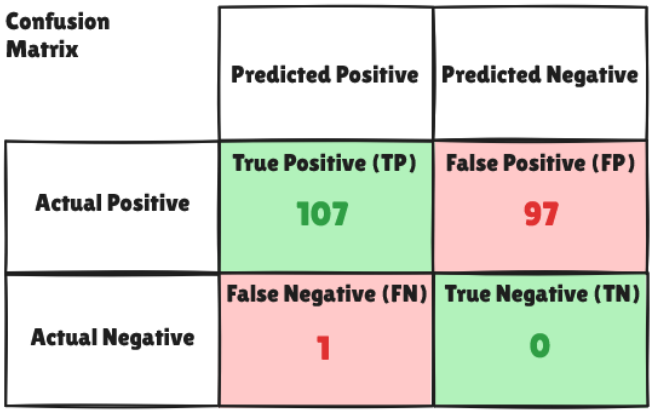


**Fig. 9.** Confusion matrix of experiment 1 (audio-transcribed redaction) with actual and predicted values of total 108 hidden PII data types. (TP + FN)

As illustrated in Figure 9, the redaction system correctly identified and masked 107 sensitive data (PII) instances, corresponding to true positives (TP), where the predicted and expected labels aligned. Only a single sensitive instance (Figure 17) was missed, resulting in one false negative (FN), which occurred for the NRP (Nationality, Religion, Political group) category. At the same time, the system produced 97 false positives (FP), largely stemming from the predefined entity labels of the employed NER models. These models tend to overgeneralize and redact less domain-specific values that are not sensitive in the given context. This behavior is particularly observed for ambiguous entity types such as locations and spoken date expressions, where contexts are limited and transcription artifacts further increase uncertainty. No true negatives (TN) are reported, as non-sensitive tokens were not explicitly annotated in the ground truth. Overall, this results in a recall of 0.99, demonstrating that nearly all sensitive entities were successfully detected, while the precision of 0.52 reflects a conservative redaction strategy that favors over-redaction. The resulting $F_1$-score of 0.69 captures this trade-off between high sensitivity and reduced selectivity.

**Original:** "The registration form is at https://forms.google.com/d/e/1FAIpQLSf for signups."

**Transcribed:** "The registration form is at HTTPS colon slash slash forms dot google dot com slash D slash E slash one FAIPPLSF for signups."

**Fig. 10.** Transcription variation through STT processing causes obfuscation.

The slight improvement over text-only results represents an unexpected finding (Figure 10) with URLs in this example being transcribed in multiple sub-strings that differ the original content.

**Secret Redaction (Experiment 2):**

Figure 11 summarizes the performance of the secret detection experiment. A total of 135 sensitive secrets were correctly identified and redacted, corresponding to true positives (TP). In contrast, 15 expected secrets were not detected, resulting in false negatives (FN), indicating cases where secret patterns were missed by the detection rules. The system also produced 22 false positives (FP), where non-secret values were incorrectly redacted, typically due to pattern-based matching that lacks full contextual awareness. No true negatives (TN) are reported, as non-secret tokens were not explicitly annotated in the ground truth.

Overall, the results demonstrate a balanced performance, with a precision of 0.86 and a recall of 0.90, most expected secrets are successfully captured. The resulting $F_1$-score of 0.88 reflects a strong trade-off between detection accuracy and generalization.

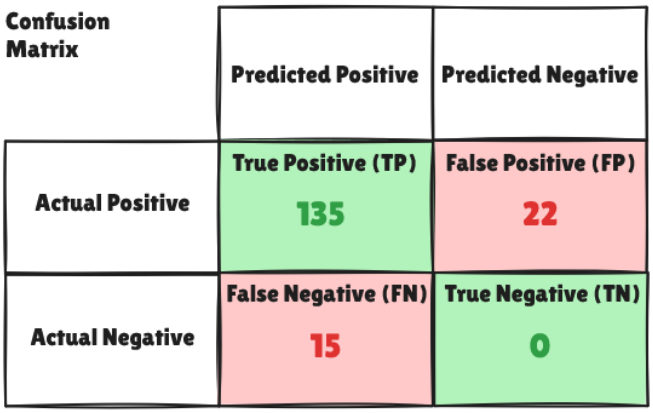


**Fig. 11.** Confusion matrix of experiment 2 with actual and predicted values of total 150 hidden secrets. (TP + FN)

| Secret Label | Detection Rate (Recall in %) |
|---|---|
| API_KEY | 41.7 |
| AWS_SECRET_KEY | 50.0 |
| GITHUB_TOKEN | 60.0 |
| AWS_ACCESS_KEY | 80.0 |
| PASSWORD | 81.8 |
| UUID | 87.5 |

**Table 8. Detection rates for underperforming secret types in Experiment 2.**

The system detected all secret data types at 100% success rate except those shown in Table 8 and a distribution of the results is displayed in Figure 18 .

**Original:** "The system administrator password is Winter2014!Tester, do not share it externally."

**Redacted:** "The system administrator password is Winter[PASSWORD]Tester, do not share it externally."

**Fig. 12.** Partially masking yields high risk for reconstruction of secrets.

Both experiments showed instances of partial redaction, where sensitive data was only partly masked (Figure 12).

## 5 Conclusion

Regarding the two initial hypotheses, the first was rejected. Audio transcription did not reduce redaction accuracy, rather introducing a slight masking effect through transcription variations. This initially counterintuitive result makes sense upon reflection: transcription errors that alter sensitive data strings, such as URL modifications (Figure 10), prevent their recognition as the original sensitive values. However, this effect is coincidental rather than a security feature, transcription variations may either help or hinder redaction depending on how they transform specific values.

The second hypothesis was confirmed, significant gaps exist in secret detection within cybersecurity and software contexts. The prototype improves detection capabilities compared to standard PII solutions, though distance from PII redaction performance remains evident.

### 5.1 Summary of Findings

**First**, label naming for detected entities represents a cosmetic concern but proves misleading for measuring actual sensitive data protection. For evaluation purposes, particularly when minimizing false negatives, generic entity labels such as [ENTITY] would provide clearer assessment than specific category names.

**Second**, the prototype confirms that customized offline redaction solutions with integrated STT transcription can be developed for domain specific deployment scenarios. The architecture demonstrates viability for privacy focused environments requiring complete local processing.

**Third**, standard NER models perform well on classical PII categories but require supporting recognizers for cybersecurity secrets. The custom regex patterns added significant detection capability but revealed fundamental limitations for high-entropy tokens lacking distinctive patterns.

**Fourth**, partially redacted values may provide enough information for reconstruction (e.g. brute-force [45]), representing a high risk challenge requiring additional mitigation strategies (Figure 12).

### 5.2 Addressing the Research Questions

**RQ1:** Offline Speech-to-Text transcription has a masking effect on PII data due to transcription accuracy variations altering original text. Transcription sometimes creates new word forms for PII data that may escape redaction. This leads to unwanted content variation from original sources, generally unintended but incidental protection. Overall, currently available offline open-source models provide nearly equivalent starting conditions for audio versus text input, resulting in nearly identical redaction accuracy. The ~2% difference between text (97%) and audio (99%) results is within those variations or called noise and does not indicate systematic advantage for either approach.

**RQ2:** Investigating secrets in cybersecurity contexts reveals increased security risk due to difficult to detect patterns like API keys and coincidental character sequences like passwords. The finding that secrets are only partially redacted in some cases indicates increasing false negative risk. In contexts where reconstruction of non-redacted parts is possible, this presents serious concerns, for example, if all characters of a cryptocurrency address except final digits could be reconstructed. Secret detection in software artifacts is achievable with notable exceptions as demonstrated. However, the limited availability of existing solutions covering only common secrets suggests that cybersecurity applications should not rely only on regex rules but rather on dedicated fine-tuned models for the specific domain.

### 5.3 Future Work and Recommendations

The potential for secret redaction in log and code files appears substantially greater than current achievements, while audio transcription quality has reached high precision with existing models. Building upon industry proven PII solutions while implementing unified labeling approaches is recommended for future development efforts.

#### *5.3.1 Enhanced sensitive data model*

As a recommendation for future based on these findings, extending an existing model such as NVIDIA GLiNER through transfer learning with dedicated labels for common or domain specific secrets represents a promising direction. Fine-tuning on carefully selected secret datasets could address the detection gaps identified for high-entropy tokens while maintaining the strong performance observed for pattern-based secrets. Such models could be trained on industry aligned formats, significantly improving detection of slightly arbitrary patterns.

#### *5.3.2 Offline desktop application*

The prototype, with implementation of the recommended enhanced model, could serve as foundation for an offline desktop application functioning as a local agent. Such an application would provide security for users of external sources such as LLMs or other internet connected services. Supporting pre-transmission redaction of sensitive data from any text or audio input. This would address the fundamental concern motivating this research: enabling beneficial AI tool usage such as external LLMs while maintaining data privacy without exposure. The containerized architecture demonstrated in this prototype provides a template for packaging such an application (e.g. with Tauri [46]) for end-user deployment.

**ACKNOWLEDGMENTS.** We thank Prof. Dr. Daniel Patrick Politze and the INS Institute for Network and Security at the Eastern Switzerland University of Applied Sciences (OST) for their valuable guidance and support during this study.

# Project Documentation

## 1 Purpose and Scope

This section documents the organizational and methodological setup of the project. It provides an overview of collaboration, timeline, risk management and tooling, and serves as a reference for understanding how the project was planned and executed.

## 2 Project Organization

### 2.1 Involved Active Stakeholders

- Etienne Kaiser (Project Team)
- Nico Heiniger (Project Team)
- Prof. Dr. Daniel Patrick Politze (Academic Advisor)

### 2.2 Collaboration

The project is carried out in close collaboration between the students, with regular coordination and feedback from the academic advisor. Planned bi-weekly meetings are held to discuss project progress, intermediate results, challenges, and next steps. Due to the explorative nature of the project, collaboration focuses on:

- Iterative refinement of the research questions
- Continuous adjustment of the technical approach
- Alignment of implementation decisions with the evaluation strategy

Design decisions, assumptions, limitations and trade-offs are documented continuously to ensure transparency and reproducibility.

### 2.3 Project Meetings

**Frequency:** Bi-weekly, with additional meetings scheduled as needed.
**Duration:** Scheduled for one hour.
**Participants:** Students and advisor.
**Purpose:**

- Review progress and intermediate results.
- Discuss technical and methodological challenges.
- Validate scope and priorities.
- Align on next steps and deliverables.

Meeting outcomes and decisions are summarized and documented.

### 2.4 Project Team Roles

As the project is conducted by a team of two with a strong focus on development, primarily backend development, no strict separation of responsibilities or fixed ownership of individual work areas is defined. This approach is chosen to reduce coordination overhead and maintain flexibility during the explorative development process. To ensure equal learning for both team members, the implementation is carried out collaboratively through close, joint work. All remaining activities, including documentation, evaluation, and project coordination, are handled as shared responsibilities of both team members. This setup reflects the exploratory nature of the project, where fast and flexible iterations and shared understanding are prioritized over strict role separation.

## 3 Project Timeline

The project follows an iterative and milestone-based timeline. CW refers to calendar weeks.

The project follows an iterative and milestone-based timeline, structured to enable early validation, continuous refinement and focused implementation efforts. The timeline is organized by calendar weeks (CW) and outlines the milestones of the project. A detailed visual representation of the planned milestones and tasks is provided in the project plan (Figure 13), which serves as the basis for scheduling and progress tracking.

#### Milestone 1: Clarification & Research (CW 38–40)

- Define research questions
- Review related work and existing solutions
- Define and document scope, gaps and experiments

#### Milestone 2: Prototype Design (CW 41–42)

- Define prototype architecture
- Select tools and identify suitable models
- Define evaluation metrics and experiments

#### Milestone 3: Implementation (CW 43–46)

- Implement audio and text processing pipeline
- Create minimal configuration and test cases

**Milestone 4: Evaluation (CW 47–48)**

- Define test datasets
- Evaluate redaction performance using precision, recall and accuracy
- Analyze limitations

**Milestone 5: Documentation & Conclusion (CW 49–51)**

- Document architecture, experiments and results
- Discuss limitations and future improvements
- Submission of the abstract
- Finalize project report

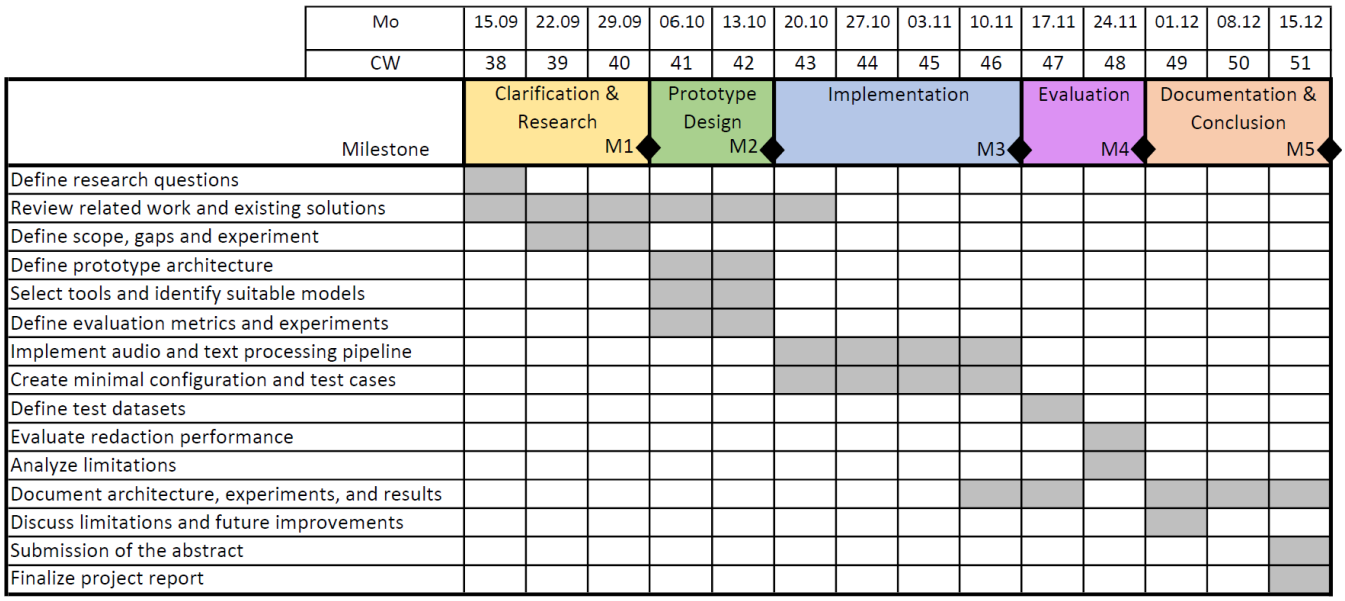


**Fig. 13.** Project Plan in Excel with the defined milestones

## 4 Risk Management

This project involves several technical and methodological risks. An overview of the identified risks is provided below, while their assessment and corresponding mitigation strategies are discussed in detail in the subsequent sections and summarized in the risk matrix (see Figure 14).

- R1 – Technology Selection Uncertainty
- R2 – Prototype Overinvestment
- R3 – Insufficient Domain Knowledge
- R4 – Unfamiliar Research-Oriented Working Style

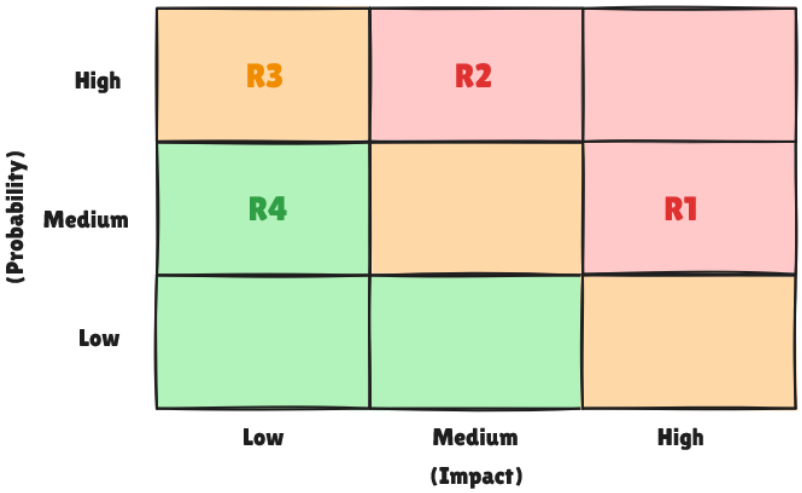


**Fig. 14.** Risk matrix summarizing identified project risks

#### R1 – Technology Selection Uncertainty

There is a risk that selecting appropriate technologies and tools may take longer than expected due to the large number of available options and uncertainty about which solutions function reliably in a local, offline setup. Incorrect or late decisions may delay implementation and experimentation.

**Mitigation:** This risk is mitigated by defining clear selection criteria early (offline capability, resource requirements, maturity, and documentation quality) and conducting small proof-of-concept tests to quickly eliminate unsuitable options without investing excessive time in each candidate. To avoid prolonged comparison phases, technology decisions are thoroughly discussed within the project team.

#### R2 – Prototype Overinvestment

There is a risk of investing too much time in the implementation, especially optimization of the prototype. As the project is explorative in nature, the primary goal is to evaluate feasibility and derive insights rather than to build a production-ready system. Both project team members tend to strive for refined solutions, which may lead to disproportionate effort spent on minor technical details.

**Mitigation:** This risk is mitigated by limiting the prototype to essential features and actively discussing the necessity and level of detail for each feature. Implementation tasks are planned early and optimization is restricted to aspects that directly support the evaluation of the research questions.

#### R3 – Insufficient Domain Knowledge

The project addresses domains such as data privacy, speech processing and redaction, where the depth of required domain knowledge may initially be underestimated. This could slow down progress or lead to incorrect assumptions, potentially affecting planning and effort estimation.

**Mitigation:** This risk is mitigated by allocating dedicated time early in the project for literature review and domain familiarization. Knowledge gaps are discussed explicitly and assumptions are validated through simple experiments or academic advisor feedback.

#### R4 – Unfamiliar Research-Oriented Working Style

The project follows a research-oriented and explorative working style, which differs from previous experience primarily focused on product-driven software development.

**Mitigation:** This risk is mitigated by slowly introducing research-oriented working practices into the project workflow. Tasks are broken down into small, well-defined investigation steps, each with a clear goal. Regular discussions with the academic advisor help to validate assumptions and ensure that the exploratory work remains aligned with academic expectations.

## 5 Tools

The following tools were used during the project. More specific AI-related tools are listed separately in Table **??** in the appendix.

- **Crixet** [47]**:** Used for writing and structuring the project documentation in an online LaTeX editor.

- **Jupyter Notebooks [43]:** Used for exploratory development, prototyping, experimentation and result analysis.
- **PyCharm [48]:** Used as the primary IDE for development, debugging and refactoring of the python [37] codebase.
- **Visual Studio Code [49]:** Used for quick edits, configuration tasks and working with various file types.
- **Docker [50]:** Used to create a reproducible local environment and manage dependencies.
- **GitHub [51]:** Used for version control, collaboration and change tracking.
- **Excalidraw [52]:** Used to create architecture sketches, workflow diagrams and conceptual illustrations.
- **Microsoft Teams [53]:** Used for communication and coordination between project team members and for meeting arrangements with the academic supervisor.
- **Microsoft Excel [54]:** Used for organizing results, creating tables, basic metric calculations and visualizing the project timeline.
- **Microsoft Word [55]:** Used for drafting documentation (either by typing or dictation) and notes prior to integration into LaTeX.
- **Diffchecker [56]:** Used to quickly compare experimental outputs and redaction results.

## 6 Use of AI Tools

- **ChatGPT [57]:** Used for language refinement, grammar correction, formulation support and brainstorming. Supplementary used for bugfixing and frontend styling.
- **DeepL [58]:** Used for producing high-quality translations and validating English formulations during the thesis writing process.
- **Google Translate [59]:** Used occasionally for quick translation checks and comparison with DeepL output to ensure correctness and consistency.

# Appendix

## A References


[1] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. DOI: 10.48550/ARXIV.2212.04356. URL: https://arxiv.org/abs/2212.04356.

[2] Alpha Cephei Inc. *Vosk Documentation*. 2025. URL: https://alphacephei.com/vosk/.

[3] Mistral AI. *Mistral AI API Documentaton*. 2025. URL: https://docs.mistral.ai/api.

[4] Nick Barney published on TechTarget. *Definition sensitive information*. 2025. URL: https://www.techtarget.com/whatis/definition/sensitive-information.

[5] SYSTRAN. *Github Repository: faster-whisper transcription with CTranslate2*. 2025. URL: https://github.com/SYSTRAN/faster-whisper.

[6] Sida Peng et al. *Transforming Redaction: How AI is Revolutionizing Data Protection*. 2024. arXiv: 2409.15308 [cs.CY]. URL: https://arxiv.org/abs/2409.15308.

[7] Martin Pelikan et al. *Enabling Differentially Private Federated Learning for Speech Recognition: Benchmarks, Adaptive Optimizers and Gradient Clipping*. 2025. arXiv: 2310.00098 [cs.LG]. URL: https://arxiv.org/abs/2310.00098.

[8] Nafiseh Soveizi, Fatih Turkmen, and Dimka Karastoyanova. "Security and privacy concerns in cloud-based scientific and business workflows: A systematic review". In: *Future Generation Computer Systems* 148 (2023), pp. 184–200. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2023.05.015. URL: https://www.sciencedirect.com/science/article/pii/S0167739X23001991.

[9] Andreas Nautsch et al. "Preserving privacy in speaker and speech characterisation". In: *Computer Speech & Language* 58 (2019), pp. 441–480. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2019.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0885230818303875.

[10] Peter Bell et al. "Adaptation Algorithms for Neural Network-Based Speech Recognition: An Overview". In: *IEEE Open Journal of Signal Processing* 2 (2021), pp. 33–66. DOI: 10.1109/OJSP.2020.3045349.

[11] Kushagra Mishra, Harsh Pagare, and Kanhaiya Sharma. "A hybrid rule-based NLP and machine learning approach for PII detection and anonymization in financial documents". In: *Scientific Reports* 15 (July 2025). DOI: 10.1038/s41598-025-04971-9.

[12] András Berzi et al. "NLP-based removal of personally identifiable information from Hungarian electronic health records". In: *Frontiers in Artificial Intelligence* 8 (May 2025). DOI: 10.3389/frai.2025.1585260.

[13] Md Nafiu Rahman et al. *Secret Breach Detection in Source Code with Large Language Models*. 2025. arXiv: 2504.18784 [cs.SE]. URL: https://arxiv.org/abs/2504.18784.

[14] Michael Meli, Matthew McNiece, and Bradley Reaves. "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories". In: Jan. 2019. DOI: 10.14722/ndss.2019.23418.

[15] Matthias Templ and Murat Sariyar. "A systematic overview on methods to protect sensitive data provided for various analyses". In: *International Journal of Information Security* 21 (Aug. 2022). DOI: 10.1007/s10207-022-00607-5.

[16] Wikimedia Foundation, Inc. *Definition Regular expression*. 2025. URL: https://en.wikipedia.org/wiki/Regular_expression.

[17] Microsoft Corporation. *Presidio: Data Protection and De-identification SDK*. 2025. URL: https://microsoft.github.io/presidio/.

[18] Amazon Web Services, Inc. *Amazon Macie: Discover and protect your sensitive data at scale*. 2025. URL: https://aws.amazon.com/macie/.

[19] Google Cloud Platform. *Cloud Data Loss Prevention (now part of Sensitive Data Protection)*. 2025. URL: https://cloud.google.com/security/products/dlp?hl=en.

[20] Google Cloud Platform. *Google Cloud Platform (GCP) Landing Page*. 2025. URL: https://cloud.google.com/?hl=en.

[21] spaCy. *Industrial-Strength Natural Language Processing Library*. 2025. URL: https://spacy.io/.

[22] NVIDIA. *NVIDIA GLiNER-PII Model Overview*. 2025. URL: https://huggingface.co/nvidia/gliner-PII.

[23] NVIDIA. *NVIDIA NeMo Overview*. 2025. URL: https://www.nvidia.com/en-us/ai-data-science/products/nemo/.

[24] NVIDIA. *NVIDIA CUDA Toolkit*. 2025. URL: https://developer.nvidia.com/cuda/toolkit.

[25] Amazon Web Services, Inc. *Amazon Transcribe Call Analytics Overview*. 2025. URL: https://aws.amazon.com/transcribe/call-analytics/.

[26] Google Cloud Platform. *Speech-to-Text API: speech recognition and transcription*. 2025. URL: https://cloud.google.com/speech-to-text.

[27] Microsoft Corporation. *What is the Speech service?* 2025. URL: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/overview.

[28] Deepgram. *The Voice AI Platform for Business Use Cases Overview*. 2025. URL: https://deepgram.com/.

[29] Speechmatics. *Speech APIs powering Voice AI Overview*. 2025. URL: https://www.speechmatics.com/.

[30] Truffle Security Co. *Trufflehog Secret Detection Overview*. 2025. URL: https://trufflesecurity.com/trufflehog.

[31] Amazon Web Services, Inc. *Amazon CodeGuru Profiler Overview*. 2025. URL: https://aws.amazon.com/codeguru/profiler/.

[32] Yelp. *Github repository: detect-secrets*. 2025. URL: https://github.com/Yelp/detect-secrets.

[33] The PostgreSQL Global Development Group. *PostgreSQL Database Landing Page*. 2025. URL: https://www.postgresql.org/.

[34] FFmpeg. *FFmpeg Developer Documentation*. 2025. URL: https://www.ffmpeg.org/.

[35] spaCy. *English pipeline (Model) optimized for CPU*. 2025. URL: https://spacy.io/models/en#en_core_web_lg.

[36] spaCy. *German pipeline (Model) optimized for CPU*. 2025. URL: https://spacy.io/models/de#de_core_news_lg.

[37] Python Software Foundation. *Python3 programming language Overview*. 2025. URL: https://www.python.org/about/.

[38] Sebastián Ramírez. *FastAPI web framework for building APIs Documentation*. 2025. URL: https://fastapi.tiangolo.com/.

[39] SQLAlchemy. *The Python SQL Toolkit and Object Relational Mapper*. 2025. URL: https://www.sqlalchemy.org/.

[40] NGINX. *nginx ("engine x") Documentation*. 2025. URL: https://nginx.org/en/docs/.

[41] OpenNMT. *Github repository: CTranslate2*. 2025. URL: https://github.com/OpenNMT/CTranslate2.

[42] iiiorg. *Huggingface Model card: Piiranha-v1*. 2025. URL: https://huggingface.co/iiiorg/piiranha-v1-detect-personal-information.

[43] LF Charities. *Jupyter Notebook Landing Page*. 2025. URL: https://jupyter.org/.

[44] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG]. URL: https://arxiv.org/abs/2010.16061.

[45] Wikimedia Foundation, Inc. *Definition Brute-force search*. 2025. URL: https://en.wikipedia.org/wiki/Brute-force_search.

[46] Tauri Contributors. *Create small, fast, secure, cross-platform applications, Landing Page*. 2025. URL: https://v2.tauri.app/.

[47] Crixet. *Crixet Online LaTeX Editor*. 2025. URL: https://app.crixet.com/.

[48] JetBrains. *PyCharm Integrated Development Environment*. 2025. URL: https://www.jetbrains.com/pycharm/.

[49] Microsoft. *Visual Studio Code*. 2025. URL: https://code.visualstudio.com/.

[50] Docker, Inc. *Docker*. 2025. URL: https://www.docker.com/.

[51] GitHub, Inc. *GitHub*. 2025. URL: https://github.com/.

[52] Excalidraw Team. *Excalidraw*. 2025. URL: https://excalidraw.com/.

[53] Microsoft Corporation. *Microsoft Teams*. 2025. URL: https://www.microsoft.com/microsoft-teams.

[54] Microsoft Corporation. *Microsoft Excel*. 2025. URL: https://www.microsoft.com/excel.

[55] Microsoft Corporation. *Microsoft Word*. 2025. URL: https://www.microsoft.com/word.

[56] Diffchecker. *Diffchecker*. 2025. URL: https://www.diffchecker.com/.

[57] OpenAI. *ChatGPT (GPT-5 / GPT-5.1)*. 2025. URL: https://chat.openai.com/.

[58] DeepL SE. *DeepL Translator*. 2025. URL: https://www.deepl.com/.

[59] Google LLC. *Google Translate*. 2025. URL: https://translate.google.com/.

## B List of Figures


1 Derived research questions from core objective. . . . 1
2 First identified research gap. . . . 4
3 Second identified research gap. . . . 4
4 Third identified research gap. . . . 4
5 Architecture and workflow of processing transcription, detection and redaction. . . . 5
6 Frontend of the prototype with comprehensive User Interface and Session Management . . . 5
7 Hypothesis for Experiment 1. . . . 6
8 Hypothesis for Experiment 2. . . . 6
9 Confusion matrix of experiment 1 (audio-transcribed redaction) with actual and predicted values of total 108 hidden PII data types. (TP + FN) . . . 8
10 Transcription variation through STT processing causes obfuscation. . . . 8
11 Confusion matrix of experiment 2 with actual and predicted values of total 150 hidden secrets. (TP + FN) . . . 8
12 Partially masking yields high risk for reconstruction of secrets. . . . 8
13 Project Plan in Excel with the defined milestones . . . 11
14 Risk matrix summarizing identified project risks . . . 12
15 Raw result comparison of labeled original ground truth (left) against final masked text (right) conducted for PII data. . . . 17
16 Results for text redaction of PII categories conducted in experiment 1. . . . 17
17 Results for audio (STT) redaction of PII categories conducted in experiment 1. . . . 18
18 Results for text redaction of secrets categories conducted in experiment 2. . . . 18


## C List of Tables


1 Comparison of text-based PII detection products. . . . 3
2 Comparison of STT-based PII detection products. . . . 3
3 Comparison of text-based secret detection products. . . . 4
4 Supported file formats by input type that can be loaded. . . . 4
5 Responsibility of used technologies in developed prototype. . . . 5
6 Test set for benchmarking of Experiment 2. . . . 6
7 Alphabetically sorted data types for test sets of both experiments. . . . 7
8 Detection rates for underperforming secret types in Experiment 2. . . . 8

## D Additional Figures

| | |
|---|---|
| Can you charge it to my credit card ending in [CREDIT_CARD]? | Can you charge it to my credit card ending in [CREDIT_CARD]? |
| I'll be there around [DATE_TIME] tomorrow afternoon. | I'll be there [DATE_TIME]. |
| Please send the invoice to [EMAIL_ADDRESS] when you get a chance. | Please send the invoice to [EMAIL] when you get a chance. |
| My IBAN is [IBAN] for the wire transfer. | My IBAN is [IBAN] for the wire transfer. |
| I'm currently living in [LOCATION] but moving to Portland next month. | I'm currently living in [LOCATION], [LOCATION] but moving to [LOCATION] [DATE_TIME]. |
| Dr. [PERSON] will be handling your case from now on. | Dr. [PERSON] will be handling your case from now on. |
| You can reach me at [PHONE_NUMBER] if there are any issues. | You can reach me at [PHONE_NUMBER] if there are any issues. |
| The documentation is available at [URL] for reference. | The documentation is available at [URL] for reference. |
| As a [NRP], I try to attend mass every Sunday morning. | As a [NRP], I try to attend mass every [DATE_TIME]. |
| Dr. [PERSON] has medical license number [MEDICAL_LICENSE] issued by the state board. | Dr. [PERSON] has medical license number MD-[ZIP_CODE]-NY issued by the state board. |
| My social security number is [SSN] for the background check. | My social security number is [SSN] for the [{entity_type}] check. |
| My driver's license number is [DRIVER_LICENSE] and it expires next year. | My driver's license number is [DRIVER_LICENSE] and it expires [DATE_TIME]. |
| I need to renew my passport before the trip - it's number [PASSPORT]. | I need to renew my passport before the trip - it's number [DRIVER_LICENSE]. |
| Please deposit the refund into account number [BANK_ACCOUNT] at Wells Fargo. | Please deposit the refund into account number [{entity_type}] at Wells Fargo. |
| My tax ID for the business is [TAX_ID] if you need it for the 1099. | My tax ID for the business is [TAX_ID] if you need it for the 1099. |
| I usually log in with username [USERNAME] but forgot my password again. | I usually log in with username [PERSON] but forgot my password again. |
| My national ID card number is [ID_CARD] for verification purposes. | My national ID card number is [{entity_type}] for verification purposes. |
| We're in the [ZIP_CODE] zip code area near downtown San Francisco. | We're in the [ZIP_CODE] zip code area near downtown [LOCATION]. |
| The payment went through on my Visa card [CREDIT_CARD] yesterday. | The payment went through on my Visa card [CREDIT_CARD] [DATE_TIME]. |
| Can we schedule the meeting for [DATE_TIME]? | Can we schedule the meeting for [DATE_TIME] at [DATE_TIME]? |
| Forward that report to [EMAIL_ADDRESS] by end of day. | Forward that report to [EMAIL] by end of day. |
| The transfer should go to IBAN [IBAN] in London. | The transfer should go to IBAN [IBAN] in [LOCATION]. |
| I grew up in [LOCATION] but now I work in New York City. | I grew up in [LOCATION] but now I work in [LOCATION]. |
| [PERSON] from accounting will handle the reimbursement. | [PERSON] from [{entity_type}] will handle the reimbursement. |
| Give me a call on [PHONE_NUMBER] when you're ready to discuss this. | Give me a call on [PHONE_NUMBER] when you're ready to discuss this. |
| Check out the tutorial at [URL] for more details. | Check out the tutorial at [URL] for more details. |
| I'm a registered [NRP] and plan to vote in the primary election. | I'm a [{entity_type}] [NRP] and plan to vote in the primary election. |
| The attending physician is Dr. [PERSON] with license [MEDICAL_LICENSE] from California. | The attending physician is Dr. [PERSON] with license [LOCATION]-MD-[ZIP_CODE] from [LOCATION]. |
| For employment verification, my SSN is [SSN] on file. | For [{entity_type}] verification, my SSN is [SSN] on file. |

**Fig. 15.** Raw result comparison of labeled original ground truth (left) against final masked text (right) conducted for PII data.

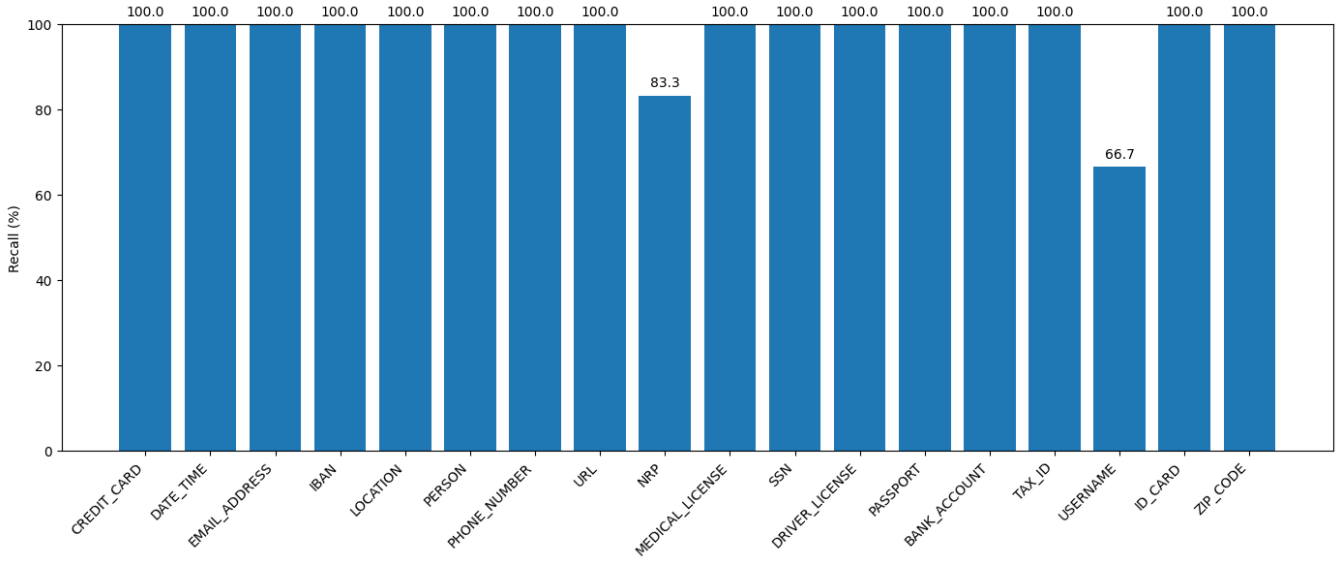


**Fig. 16.** Results for text redaction of PII categories conducted in experiment 1.

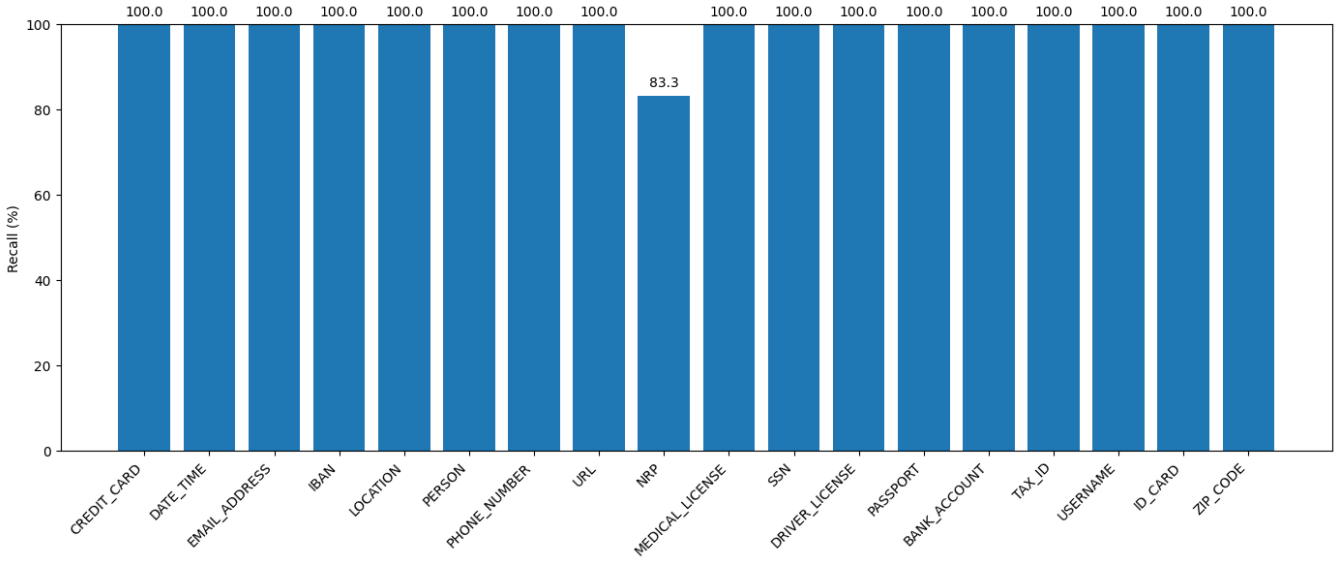


**Fig. 17.** Results for audio (STT) redaction of PII categories conducted in experiment 1.

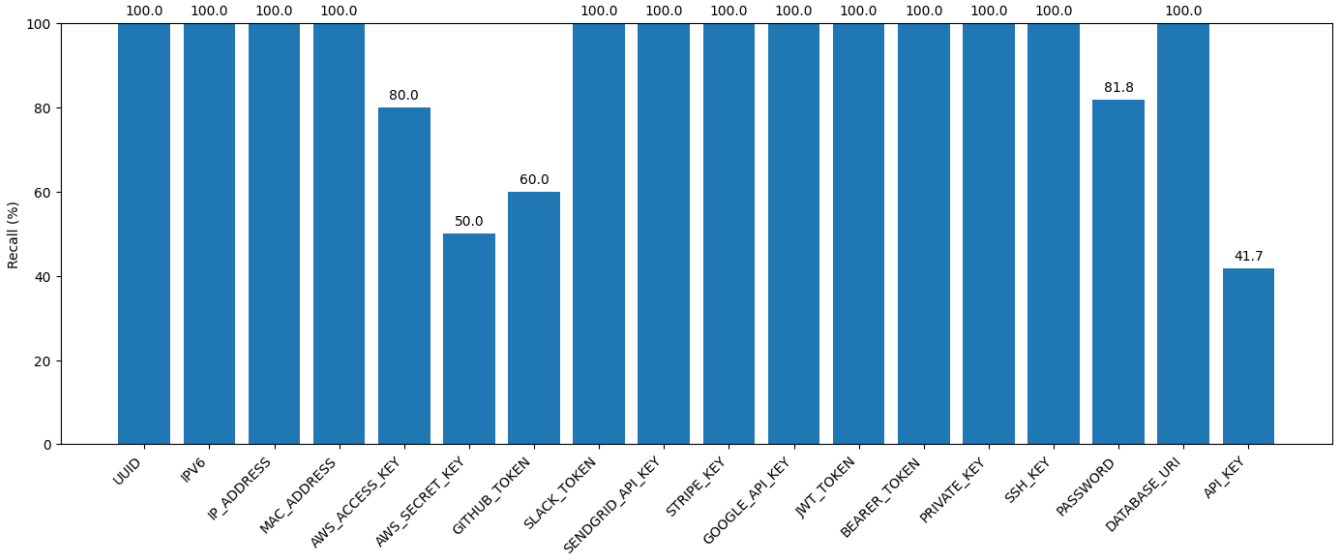


**Fig. 18.** Results for text redaction of secrets categories conducted in experiment 2.