Universiteit
Leiden
The Netherlands

Data Science & Artificial Intelligence

Extraction of metadata from debt letters:
Comparison of local NLP models

Beyza Celep

First supervisor: Joost Visser
Second supervisor: Natalia Amat Lefort

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

15/07/2025

Abstract

Background: Debt collection letters are often difficult to understand, especially for individuals in financial distress. These letters contain critical metadata such as the amount owed, due date, and creditor identity. However, they are often unstructured, written in complex legal language, and received as noisy OCR scanned documents. Automatically extracting this metadata is necessary to enable downstream services and reduce manual workloads, particularly in privacy-sensitive domains where cloud-based solutions are unsuitable.

Aim: This thesis investigates whether local Natural Language Processing (NLP) models can effectively extract structured metadata from noisy Dutch debt collection letters. Specifically, the study compares three approaches: a feature-engineered Conditional Random Field (CRF), a fine-tuned SpaCy Named Entity Recognizer (NER), and instruction-following local Large Language Models (LLM).

Method: A realistic synthetic dataset of 1,000 debt letters was generated, augmented with OCR noise, and annotated programmatically. Each model was either explicitly trained (CRF, SpaCy) or instructed via the same prompt (LLMs) to extract the metadata fields and evaluated using precision, recall, F1-score, and execution time.

Results: CRFs provided reliable extraction of predictable fields. SpaCy performed well on common entities but failed on others. The local LLM achieved the highest field coverage but incurred higher computational costs.

Conclusion: In privacy-constrained legal-financial settings, instruction-tuned local LLMs offer promising performance, but lightweight models remain competitive for only structured tasks.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Joost Visser, for his continuous support and guidance throughout this thesis. Even in the moments that I struggled, showing a clear direction his thoughtful feedback and encouragement helped me stay on course. I greatly appreciated the balance he maintained and giving me the freedom to explore independently, yet always offering the right insights when I needed them most.

I also want to thank my parents for always loving, being patient, and supporting me. They are the reason I have been able to do everything I have accomplished. Their unwavering support and faith in me have helped me get through the most difficult and rewarding parts of this journey.

Thank you!

Contents

Chapter 1

Introduction

Debt collection letters are important financial documents delivered to individuals and organizations to inform them of their outstanding financial obligations. This is a crucial aspect of public service in the Netherlands. The National Tax Service (Belastingdienst) issues around 1.8 million tax assessments for businesses and a comparable number of income tax determinations for individual citizens annually. In addition, the Central Judicial Collection Agency (Centraal Justitieel Incassobureau, (CJIB)) issues about 8 million fine notifications. The high volume of such communications underscores how common and impactful debt letters are in the daily lives of Dutch residents. However, many of these letters are difficult to process, not just for recipients, but also for the institutions that handle them. Their unstructured nature, legal jargon, and inconsistent formatting create significant barriers to both human and automated understanding.

A debt collection letter generally contains various details such as; the debtor's identity, the creditor or managing agency, the total amount owed, essential dates, and multiple reference identifiers. Upon receiving this information, the recipient has to determine the outstanding amount, the deadline for payment, and the action that needs to be taken. Frequently necessitating further communication with the issuer to dispute charges or to negotiate a repayment plan.

Many individuals who receive these letters find them difficult to comprehend or even intimidating, particularly as they contain formal legal terminology or several figures and regulations. Misinterpretation generally leads to postponing repayment, intensifying financial struggles, or resulting in unwarranted penalties. For those seeking to restore financial stability, the initial stage typically involves interpreting these letters, collecting essential metadata, organizing it, and formulating a planned response.

AI-supported automated systems for data extraction and financial recommendation can be transformative in addressing these challenges. These technologies are capable of accurately identifying and extracting critical information from scanned or digital documents, including creditor names, payment deadlines, amounts due, and legal references. By transforming unstructured debt letters into structured data, they facilitate the faster processing, more accurate interpretation, and more informed decision-making of both individuals and support organizations. To further assist recipients, financial recommendation components can suggest customized repayment strategies or emphasize imperative actions. Not only do these intelligent systems ease the administrative burden on caseworkers and municipal services, but they also provide individuals with a more comprehensive understanding of their financial obligations. They can assist in bridging the distance between complex bureaucracy and citizen comprehension when implemented ethically and transparently, thereby transforming the debt recovery process into one that is more accessible, timely, and humane.

1.1 Problem Statement

Despite the importance, debt letters typically come in a wide range of forms, languages, and layouts. This is especially the case in the Netherlands, where each creditor may use a distinct template. Automatic information extraction is needed but challenging since there is no standardization, the legal language is hard to understand, and scanned documents often have OCR problems.

Finding debtor names, reference numbers, and due dates in these letters automatically is a first step in making workflows easier for municipalities, financial service providers, and legal advisers. But it is still an open technical problem to come up with a solution that is both accurate and resource-efficient, and that does not depend on cloud services (due to privacy issues).

These issues are compounded by the document variability and technical limitations in OCR processing, which make metadata extraction a particularly difficult task.

1.2 Research Aim

The goal of this thesis is to find out if lightweight, privacy-preserving natural language processing models can consistently pull out important metadata fields from debt collection letters that are noisy and only partially organized. In particular, it investigates the pros and cons of using traditional machine learning approaches (such as Conditional Random Fields), newer neural architectures (like SpaCy NER), and new instruction-following large language models (LLMs) that are deployed locally.

We evaluate each model on both technical metrics (accuracy, F1-score, recall) and practical concerns (runtime on local hardware, memory usage, explainability). This dual perspective ensures the solutions are not only performant but also deployable.

This thesis addresses a notable gap in the field of privacy-preserving natural language processing by focusing on metadata extraction from debt collection letters, an understudied yet socially impactful domain. Specifically, it compares three approaches: Conditional Random Fields (CRF), SpaCy's statistical NER pipeline, and instruction following local large language models (LLMs), all executed in a fully local environment. While prior research has explored named entity recognition in legal or financial texts, most rely on cloud-based LLMs or general-purpose models without considering the privacy risks or deployment limitations inherent to sensitive personal data. By assembling a dedicated corpus of debt letters and evaluating these three techniques under the same conditions, this work contributes a novel empirical benchmark for real-world, privacy-respecting document processing in the legal-financial domain.

1.3 Research Questions

The main question guiding this thesis is:

> **RQ:** Which local Natural Language Processing model is most suitable for extracting structured metadata from Dutch debt collection letters, considering accuracy, robustness to noise, computational efficiency, and deployment feasibility?

To follow up the main research question the following sub-questions are addressed:

- **RQ1:** How well does a rule-enhanced statistical model such as Conditional Random Field model work when it comes to structured field extraction from OCR-processed debt letters?
- **RQ2:** Can SpaCy's fine-tuned NER pipeline work with synthetic Dutch debt letters, and which fields does it do well or poorly?
- **RQ3:** How well does a locally deployed instruction-tuned LLM work for extracting metadata?
- **RQ4:** What are the costs, inference times, and resource needs of each method when using consumer-grade hardware?
- **RQ5:** How do the models stack up against each other when it comes to being scalable, and easy to add to local document processing workflows?

1.4 Thesis Structure

This thesis is structured as follows. Chapter 2 reviews the evolution of metadata extraction techniques, the challenges specific to debt letters, and recent trends in NLP and document understanding. Chapter 3 discusses how a synthetic dataset was generated, how it was annotated, and how each of the three extraction pipelines was set up. Chapter 4 demonstrates a quantitative comparison of the models based on their accuracy, recall, F1-score, and runtime. Chapter 5 discusses the results in light of real-world

factors including privacy, deployment costs, and how many mistakes they can handle. Finally Chapter 6 sums up the results, addresses the study questions, and suggests future work.

Chapter 2

Background and related work

This chapter provides the theoretical foundation for the metadata extraction task explored in this thesis. It begins by defining metadata and discussing the evolution of metadata extraction techniques, particularly in the context of OCR-processed documents such as PDFs. It then introduces a range of methods used in prior work, including rule-based systems, CRFs, statistical named entity recognition using SpaCy, and transformer-based and instruction-following language models. These techniques are examined in relation to their applicability for extracting information from unstructured financial correspondence, setting the stage for the comparative analysis of their performance in the following chapters.

2.1 Metadata Extraction

Metadata is defined as ‘‘ data that provides information about other data ’’ serving as a contextual layer that outlines the characteristics, structure, and management of a data resource, excluding its actual content, such as the text of a document or the pixels of an image [4]. It allows the classification, recognition, and effective administration of digital assets across several domains. Metadata can be in many different formats based on its intended use. Descriptive metadata facilitates information retrieval and identification, including titles, authorship, and keywords. Structural metadata defines the arrangement of a resource's components, such as chapters in a book or the layout of a PDF. Administrative metadata facilitates resource management by supplying information about file type, creation date, access permissions, and preservation. Collectively, these categories constitute the basis of digital resource organization and automation.

Metadata extraction involves finding and obtaining structured information from unstructured or semi-structured materials. These assets generally include several forms of digital content, including webpages, photos, audio files, and PDF documents. Metadata extraction facilitates automated document processing, retrieval, indexing, and management by transforming implicit knowledge into explicit structured formats [3]. Early metadata extraction methods to retrieve textual materials depended on conventional rule-based and lexical heuristic techniques. These methodologies utilized patterns, keywords, and formatting structures to identify and extract essential information. Nevertheless, they relied significantly on manual configuration and specialized knowledge, which constrained their scalability and adaptability across various document types [6].

Improvements in machine learning, especially statistical models such as CRF, significantly improved the precision and flexibility of metadata extraction from text. CRF's enhanced the capacity to recognize context and model sequential dependencies, enabling systems to flexibly adjust to structural changes in documents without requiring explicit rules [7]. Recently, deep learning methodologies particularly transformer-based models such as BERT, LayoutLM, and Longformer have significantly progressed the field [15]. These models, trained on extensive and varied datasets, adeptly capture contextual, semantic, and structural connections within documents, leading to significant enhancements in generalization and extraction accuracy.

A particularly challenging document format is the PDF, extensively used in fields such as law, finance, and administration. Despite being widely used, PDFs are challenging to process due to their intricate

and frequently inconsistent formatting [11]. Metadata extraction from PDFs generally begins with Optical Character Recognition (OCR), which transforms scanned images or non-machine-readable text into machine-readable digital formats. Still, OCR presents difficulties, including incorrect recognition and segmentation, particularly when confronted with low-quality scans or visually complex patterns.

Post OCR, metadata extraction often utilizes hybrid models that integrate heuristic preprocessing techniques, including regex matching and keyword identification, with statistical or deep learning-based extraction methods [10]. These hybrid methods are especially effective for papers that have somewhat predictable yet inconsistent formats, including bills, receipts, and legal notices.

Debt collection letters are a perfect illustration of the intricacy of metadata extraction. These documents include critical legal and financial metadata, including debtor and creditor identities, outstanding amounts, due dates, and case identifiers, which have to be precisely retrieved for subsequent automated processing. In contrast to standardized forms, debt letters frequently exhibit significant variation in structure, language, and layout. Given this unpredictability and OCR failures, it is necessary to have strong extraction techniques that can handle a variety of text formats and noise. The successful extraction of metadata from these letters is essential for optimizing financial procedures, facilitating automation, and maintaining regulatory compliance.

2.2 Techniques for Metadata Extraction

A wide range of NLP approaches have been explored for document metadata extraction, evolving over time from rule-based systems to deep learning. Here we summarize the major categories and situate our chosen methods:

2.2.1 Rule-Based and Lexical Heuristics

Early or low tech solutions rely on manually crafted rules or regular expressions to find patterns. For example, a sequence of digits that matches an invoice number format or detecting the word “Dear” followed by a capitalized name as the debtor’s name. While straightforward and highly precise for simple cases, rule-based methods struggle with the diversity of languages and require significant maintenance for each format variation [17]. Given the complexity of debt letters, pure rule-based extraction would be weak. However, some rule heuristics can be incorporated into more robust models as features or post processing.

2.2.2 Conditional Random Fields (CRFs)

CRFs are a class of statistical sequence models that have been very successful in named entity recognition (NER) and similar tasks in NLP. A CRF models the conditional probability of a label sequence given an input sequence of tokens. Essentially, it assigns labels such as “DEBTOR NAME” or “O” for non entity to each token in a sequence by considering not only the token’s features but also the context of neighboring labels [7]. Compared to generative sequence models, CRFs are discriminative and can incorporate arbitrary, overlapping features of the input without making independence assumptions. For example, one can define features like “token contains digits”, “token is capitalized”, or “previous token is "Invoice", and the CRF will learn how these features correlate with particular entity labels. In documents, CRFs have been used to extract fields like addresses and dates by exploiting consistent local patterns for example an address might be detected by a pattern “Number + Capitalized Word” meaning street number and name.

In contrast, Hidden Markov Models (HMMs) are generative models that define joint probabilities over both the observed data (e.g., words in a sentence) and the hidden label sequences (e.g., entity tags). They rely on the assumption that each token’s label depends only on the previous label, and that the observed word depends only on the current label. This independence assumption limits flexibility in incorporating overlapping or arbitrary features, which CRFs overcome.

Figure 2.1 offers a visual comparison of CRFs and related probabilistic models. It shows how CRFs generalize logistic regression and sequence models like HMMs by modeling the conditional dependencies among output labels, making them well-suited for structured prediction in text.

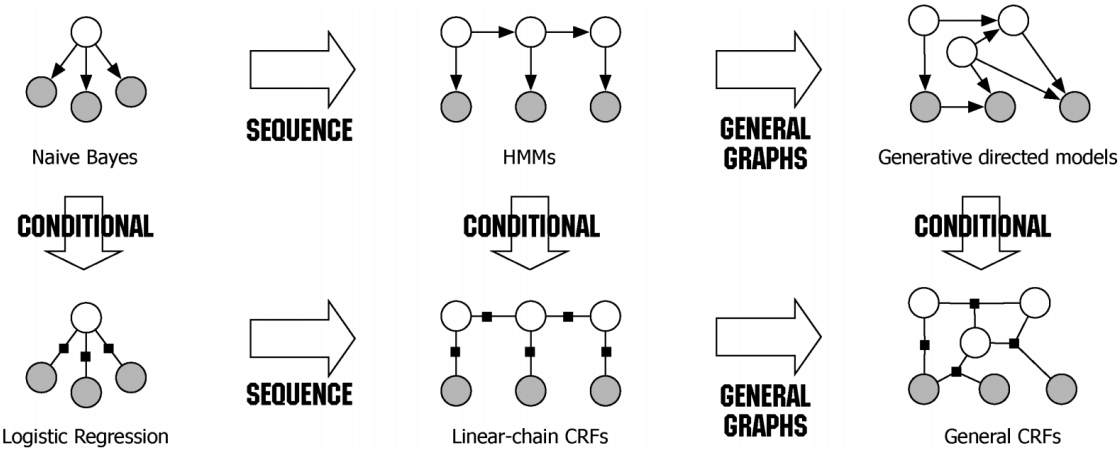

Figure 2.1: Evolution of probabilistic models for sequence and structured prediction. Conditional Random Fields (bottom row) extend logistic regression to structured outputs by modeling dependencies between output labels. Linear-chain CRFs (center) are particularly effective for tasks like Named Entity Recognition (NER). Image taken from Sutton and McCallum [12].

CRF's are relatively lightweight in terms of computation and can be made very precise with good feature engineering. A big advantage is explainability, since features are human defined, one can often trace why the model made a decision for example it labeled "Amsterdam" as a city because the feature word is city was true. CRFs require a labeled training set for supervision, and performance can degrade if the test data distribution differs significantly from training for example a new letter layout might confuse it if features were very layout specific [17]. Despite the rise of deep learning, CRFs remain in use for domains where data is limited but expert knowledge can be encoded as features.

Figure 2.2 illustrates how CRFs assign a label to each token based on handcrafted features and context.

Input: Mr. John Smith paid €500

Tokens: Mr. John Smith paid €500
Features: [Title] [Cap] [Cap] [Verb] [Currency]
↓
CRF Sequence Model
↓
Labels: O B-PER I-PER O B-AMOUNT

Figure 2.2: Simplified view of CRF-based sequence labeling. Each token is represented by a set of handcrafted features, and the CRF predicts a label sequence.

2.2.3 Statistical NER with Local Models (SpaCy)

Modern NLP libraries like SpaCy provide ready to use NER components that are typically powered by neural networks. SpaCy's NER, for instance, uses a convolutional neural network (CNN) architecture with word embeddings and context-sensitive representations to identify entity spans in text. Instead of labeling each token independently, these models learn to recognize multi token spans by training on annotated examples such as an organization name that is multiple words [2]. They do not rely on manually defined regex or token level flags, but rather learn internal features (embeddings) that encode semantic and shape information. For example, SpaCy's model will learn the character patterns and contexts that suggest a token is a name or an address, guided by training data.

A key benefit is that once the model architecture is set, adding a new entity type like "DEBT AMOUNT" just requires providing examples of that entity in context. The model will adjust its weights to capture the new concept. This reduces the need for manual feature engineering which highlights SpaCy's well

performance on messy input as it generalizes phrases rather than just individual token tags. However, purely statistical models can make mistakes that are hard to interpret, and they may require a relatively large number of training examples to learn patterns that a human could specify via rules. Fine-tuning SpaCy on a new domain in this paper's case Dutch debt letters involves creating a labeled dataset and possibly incorporating some domain knowledge. SpaCy does allow adding custom pipeline components or matchers for specific patterns.

Figure 2.3 illustrates SpaCy's model pipeline, which learns to extract multi-token spans through training rather than relying on predefined token-level features.

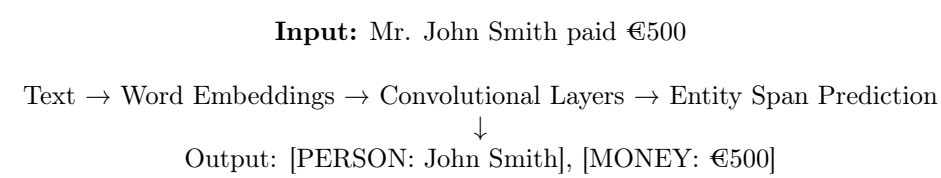

Figure 2.3: Simplified SpaCy-style pipeline for NER. Unlike CRFs, the model learns embeddings and span representations automatically from training data.

In related work, domain-specific NER has been achieved by training SpaCy or similar models on synthetic data or augmenting real data for example, SpaCy has been successfully adapted for medical NER and other specialized tasks by providing it with the right training corpus. In this paper, we can leverage SpaCy's pretrained Dutch language model as a starting point, so that general language structure is known, and then fine-tune it with our letter data to learn the custom entity categories.

2.2.4 Transformer-Based Models

The last decade has seen transformers and pre-trained language models revolutionize NLP, as explained by Xu et al. [16]. Models like BERT and RoBERTa introduced contextual embeddings that significantly improved NER performance by capturing long-range dependencies in text. A BERT fine-tuned for NER can classify each token into an entity category with high accuracy, often outperforming CRF-based models in general domains. However, vanilla transformer models treat the text as a sequence of tokens without any notion of document layout. In tasks like form or letter understanding, layout matters e.g., the position of an address block or the fact that an item is in a header vs body can be an important clue. To address this, layout-aware transformers have been developed. LayoutLM is one notable model that incorporates the 2D position of text in the document image along with the text content into a unified transformer representation. This allows it to learn, for example, that text at the top of a letter in bold might be a header or that an address usually appears as a block in the top-left, etc. LayoutLM and its successors (LayoutLMv2, etc.) have achieved state-of-the-art results on tasks like form understanding and receipt extraction by jointly modeling text and layout.

Other approaches like Donut (Document Understanding Transformer) take this further by being OCR-free, handling the image of the document in an end-to-end. Donut essentially combines vision and language understanding: it can directly parse an image of a document without an intermediate OCR step, which avoids OCR errors and can be more flexible with different scripts or noisy scans. These transformer approaches, while powerful, usually require considerable computational resources (especially if fine-tuning them on custom data) and large amounts of training data to realize their full potential [5]. In a corporate setting, one might not have the liberty to deploy a 300 million parameter model on every user's machine or edge device.

2.2.5 Large Language Models (LLMs)

The latest LLMs (GPT-4, Claude, etc.) are extremely general and can perform extraction via prompting. For instance, feeding the text of a letter to GPT-4 with a prompt such as: "Extract the debtor's name, debtor's address, creditor's name, amount, and date from the letter below and output as JSON." and the model will usually comply, since it has learned to follow instructions and format outputs. Studies have

shown LLMs can achieve high accuracy on information extraction in zero-shot or few-shot settings, even for specialized domains [1]. However, using such models in practice raises issues: privacy (as discussed, data must not leave the local environment unless one hosts the model), cost, and consistency. LLMs can sometimes hallucinate e.g. if the letter is ambiguous or an OCR error garbles a field, a generative model might make up a plausible value or misinterpret the request [13]. This unpredictability is risky in a product context where wrong outputs could have legal implications.

There is a growing interest in local deployment of LLMs to mitigate privacy and control issues. Open-source models (like LLaMA 2, GPT-J, etc.) can be run on local servers or even high-end laptops, though typically the largest models (70B+ parameters) need specialized hardware Wiest et al. [14]. demonstrated that a locally deployed LLM (LLaMA 2 70B) achieved very high accuracy in extracting clinical information, rivaling cloud-based solutions, all while maintaining privacy. This suggests that choosing an appropriately sized model, a local LLM might indeed serve as a strong baseline or competitor to our traditional NLP models, without compromising data privacy. The question remains whether a small model (such as 1B parameters) can achieve acceptable accuracy on debt letters. LLMs also generally require more memory and are slower than task-specific models.

In summary, related work spans from CRFs and rule-based extractors that were common in the 2000s-2010s, to neural NER systems (like SpaCy's) in the late 2010s, to transformers and LLMs in the 2020s that currently lead many benchmarks. This thesis is inspired by the need to balance state-of-the-art techniques with real-world constraints. In this research the approaches that ensure data never leaves the organization and that can run on modest computing infrastructure are prioritized. This is in line with a privacy-first, product-driven research philosophy: prefer solutions that are explainable, cost-efficient, and integrate easily into existing workflows.

Chapter 3

Method

The research methodology consists of (1) constructing a synthetic yet realistic dataset of debt letters with ground truth annotations, (2) implementing three extraction pipelines (CRF-based, SpaCy NER, and local LLM) using that data, and (3) defining evaluation procedures focusing on accuracy, speed, resource use, and integration effort.

3.1 Challenges in Debt Letter Metadata Information Extraction

Debt collection letters pose unique challenges for information extraction due to their format and language. Unlike structured forms or invoices, these letters are often written in free form prose with varying layouts. There is no universal template; each creditor or agency may use a different document structure, and even within one organization the letters can vary over time. Key details like the debtor's name, the creditor's account number, or the total amount due might appear in different positions or contexts in each letter. Some letters prominently list a table of details at the top, while others bury these details in paragraphs of explanation. The text itself tends to be long and complex, mixing formal legal language with financial calculations and instructions. Recipients often find these letters "intimidating and confusing to read.

Despite this variability, most debt letters do contain a common set of metadata fields that are crucial to extract [9]. These typically include:

- **Subject:** A brief line indicating the subject of the letter (e.g. "Subject: Second Reminder").
- **Debtor Identity:** Name of the person or organization that owes the debt (often appearing in the letter header or salutation).
- **Debtor Contact Details:** Especially the postal address of the debtor, and sometimes other contact information.
- **Creditor Identity:** Name of the entity to whom the debt is owed (which may be a different current owner if the debt was sold.
- **Creditor Contact/Account**: Details for the creditor such as their address or an account number (e.g. bank IBAN) where payments should be sent.
- **Debt Amount:** The principal amount of debt owed, sometimes including interest or fees.
- **Dates:** Relevant dates, for example the date of the letter, the invoice date, or a payment due date.
- **Reference Numbers**: Identifiers like a case file number, client number, invoice number, or letter ID used to reference this debt in records.

These fields allow any automated system to understand who the letter is about, who is demanding payment, how much is owed, and key reference info. Prior work in document analysis and compliance has highlighted including such elements clearly in debt notices. Any solution for extracting information from debt letters must reliably identify these fields, despite variations in wording (e.g. "outstanding amount" vs "remaining balance") and formatting.

Another challenge is that many debt letters are processed as scanned documents. Agencies often send letters via postal mail, so the digital pipeline may involve scanning a printed letter to PDF and running OCR (Optical Character Recognition) to get machine-readable text. OCR can introduce errors such as misreading digits or letters, especially under noise or low-resolution conditions. Thus, extraction models may have to cope with noisy input text containing typos or odd spacing from OCR [8]. Advanced techniques like layout-aware models (discussed below) can use the positional layout of text to improve accuracy, but in a fully local setting they may be too resource intensive. Our approach takes OCR text as input and focuses on making the NLP robust to noise through preprocessing and domain specific tuning.

To address these challenges in a controlled and privacy-preserving way, a synthetic dataset designed to reflect the variability and noise found in actual debt letters is constructed.

3.2 Realistic Synthetic Dataset Construction and Augmentation

Due to the sensitivity and limited availability of authentic real debt collection letters, a synthetic dataset was constructed to closely replicate the content, structure, and visual appearance of real Dutch debt letters. The data generation process is automated using a custom Python script, `debt_letter_generator.py`, which facilitates the creation of a large volume of diverse and realistic instances.

1. Base Template and Content Insertion A set of letter templates was created to reflect the standard layout and tone commonly used by Dutch collection agencies. Each template includes a header (containing agency address and logo placeholder), salutation, main body, and footer. Realistic values are inserted into these templates using the `Faker` library, configured for the Dutch locale (`nl_NL`), enabling:

- Generation of debtor names and addresses
- Selection of creditor names from a predefined list
- Randomized assignment of plausible dates and monetary amounts
- Insertion of legal boilerplate text sourced from a curated library

This ensures that each letter maintains the formal tone and verbosity typically found in official correspondence. For example, debtor details are generated as follows:

```
debtor_name = fake_nl.name()
debtor_address = fake_nl.address()
```

2. PDF Formatting and Rendering The `ReportLab` library is used to render the populated templates into styled PDF documents. Layout features such as fonts (e.g., `Helvetica`, `Courier`), text alignment, spacing, and address block positioning are varied to simulate different letterhead styles. Elements such as reference numbers or QR codes are included, with QR codes generated using:

```
img_buffer = create_qr_code(letter_data["letter_id"])
```

This stage results in a clean, high-resolution PDF representing a digital version of the debt letter.

3. Data Augmentation via Simulated Scanning and OCR Noise To emulate real-world conditions where debt letters are frequently received as scanned copies data augmentation is applied through visual distortions. These transformations preserve the semantics while introducing visual variability by:

- Downsampling to lower resolutions
- Application of Gaussian noise and blur
- Slight page rotation or skew
- Simulation of font degradation effects

Distorted images are subsequently processed using the Tesseract OCR engine to extract text content:

```
ocr_text = pytesseract.image_to_string(distorted_image)
```

The OCR output contains typical recognition errors, such as misreading ‘2’ as ‘Z’ or interpreting ‘€104.02’ as ‘€104.0Z’. These imperfections create realistic test conditions for assessing the robustness of NLP extraction methods.

4. Annotation Generation Since all inserted content is programmatically generated, ground truth labels for each field are retained. A structured JSON annotation is saved for every letter, mapping each field name to its corresponding value. An example annotation is shown below:

```
{
  "debtor_name": "Jente Rousselet-Verhoeven",
  "debtor_address": "Luukbaan 748, 4837 YU Schoonloo",
  "creditor_name": "Deurwaarders Collectief",
  "creditor_account": "NL28KZGR8946401286",
  "debt_amount": "€104.02",
  "letter_date": "26-09-2024",
  "letter_id": "1e002886-31c4-420f-b63c-3ea5abc7bb58",
  "subject": "Tweede Aanmaning"
  ...
}
```

These annotations facilitate supervised training and allow precise evaluation by comparing model predictions to ground truth values. Field names are standardized across examples.

5. Final Dataset Structure Each synthetic data point comprises a visually distorted PDF image and a corresponding JSON annotation. Approximately 1,000 letters were generated, with 800 used for training and 200 held out for testing. Templates are shared across splits to assess generalization. The randomized content ensures the models are required to generalize to unseen values.

6. Annotation Quality and Validation Although annotations were generated automatically from inserted values, a manual inspection was performed on 50 randomly selected letters. Each annotated field was compared against the visual and textual content, with a correction rate below 2%. These checks confirmed the annotation process was sufficiently reliable for supervised training and evaluation.

3.3 OCR Preprocessing and Text Cleaning

Before applying NLP models, all scanned debt letters are passed through an OCR pipeline using Tesseract. The output is then cleaned to reduce noise by normalizing common OCR typos (e.g. ’0’ vs ’O’), removing extra whitespace, fixing broken tokens using heuristics, and splitting paragraphs into sentences to help downstream token-level models perform better. This preprocessing ensures the input to SpaCy, CRF, and LLM models is as consistent and readable as possible.

3.4 SpaCy NER Pipeline for Metadata Extraction

To extract structured metadata from OCR-processed Dutch debt collection letters, a Named Entity Recognition (NER) pipeline was developed using SpaCy. This pipeline is implemented in `debt_ner.py` and is designed to identify key information fields such as names, amounts, dates, and account numbers directly from noisy letter text. The goal of this pipeline is to produce structured outputs that match the fields in the original annotation JSON, enabling automated parsing of scanned financial correspondence.

1. Label Definitions and Mapping The first step involves defining a fixed set of entity labels that the model should learn to detect. These labels correspond to the metadata fields annotated in the JSON files, such as `"debtor_name"`, `"debt_amount"`, and `"letter_date"`. These field names are mapped to SpaCy-compatible uppercase entity labels, such as `DEBT_AMOUNT` or `CREDITOR_ACCOUNT`:

```
LABELS = [
  "DEBT_AMOUNT", "ONDERWERP", "ORIGINAL_AMOUNT", "LETTER_DATE",
  "DEBTOR_NAME", "DEBTOR_ADDRESS", "CLIENT_NUMBER", "FACTUUR_NUMMER",
  "FACTUUR_DATUM", "VERVAL_DATUM", "CREDITOR_NAME", "CREDITOR_ADDRESS",
```

```
  "CREDITOR_ACCOUNT", "CREDITOR_EMAIL", "LETTER_ID"
]
KEY2LABEL = {
  "debt_amount": "DEBT_AMOUNT",
  "Onderwerp": "ONDERWERP",
  ...
}
```

This is a crucial step as SpaCy requires entity labels to be explicitly declared and used consistently throughout the training data. By mapping human-readable field names to standardized NER labels, the pipeline ensures compatibility with SpaCy's internal training and evaluation mechanisms. This also supports clear alignment between the annotations and model outputs.

2. PDF Text Extraction To retrieve raw text for model input, the function `pdf_to_text()` uses the `pdfplumber` library to extract text from each page of a PDF. It avoids OCR at this stage, relying on previously embedded text from synthetic generation:

```
with pdfplumber.open(str(path)) as pdf:
    for page in pdf.pages:
        text = page.extract_text() or ""
```

This is important for accurate text extraction as it is essential due to any mismatch between the letter content and the annotated fields will compromise the ability to train the model correctly. This function ensures reliable, page-wise extraction of text for each letter.

The function `find_span()` is responsible for locating the character offsets of each annotated value within the extracted text. It first attempts an exact match using `text.find()`, and if that fails (often due to OCR noise), it applies fuzzy string matching via the `rapidfuzz` library:

```
cand, score, loc = process.extractOne(
  clean(value), [(m.group(), m.start()) for m in re.finditer(r".{20}", text, re.S)],
  scorer=fuzz.token_sort_ratio,
)
```

This is important as OCR degradation often introduces subtle errors such as character substitutions or spacing artifacts. Fuzzy matching ensures that even imperfect renderings of the original field values can be aligned with their location in the text, allowing training to proceed with noisy inputs.

3. Training Data Construction Training examples are constructed by loading the text from each PDF and its corresponding JSON annotation. The script attempts to find each field's text span and converts valid spans into `char_span()` objects in a SpaCy Doc. These spans are compiled into example objects:

```
span = doc.char_span(start, end, label=KEY2LABEL[k], alignment_mode="contract")
example = Example.from_dict(doc, {"entities": [span for span, _ in spans]})
```

Longer spans are prioritized to avoid overlap with shorter substrings (e.g., full names vs. partial names).

This makes sure supervised NER training as SpaCy requires span-annotated documents. This process ensures that each entity is correctly located within the letter's text, enabling the model to learn to recognize and generalize these patterns during training.

4. NER Model Training The Dutch language model `nl_core_news_lg` is loaded as the base and extended with the custom NER component. During training, examples are shuffled and passed to SpaCy's training loop, where weights are updated using backpropagation:

```
for i in range(n_iter):
    random.shuffle(train_examples)
    losses = nlp.update(train_examples, drop=0.35)
```

Fine-tuning an existing language model allows leveraging pre-trained linguistic features while adapting the model to the specific domain of debt communication. This is more efficient than training from scratch and leads to better performance on limited data.

5. Inference and Field Recovery At inference time, a letter's OCR text is passed through the trained model. Detected entities are mapped to their original keys and stored in a result dictionary:

```
doc = nlp(ocr_text)
results = {ent.label_: ent.text for ent in doc.ents}
```

If multiple entities of the same label are found, a selection strategy (e.g., choosing the first occurrence) is used. Inference transforms noisy unstructured text into a clean, structured metadata dictionary. This is the core functionality needed to automate letter processing in real-world applications.

3.5 CRF-Based Metadata Extraction Model

A Conditional Random Field (CRF) model was developed to extract structured metadata from OCR-processed debt collection letters. This model treats the letter text as a sequence of tokens and assigns each token a label corresponding to a metadata field or the label "O" (outside any entity). The implementation is encapsulated in the `DebtLetterCRF` class and utilizes the `sklearn-crfsuite` library for training and inference.

1. Labeling Scheme and Model Configuration Each token is annotated using the BIO tagging scheme (Begin, Inside, Outside), which allows for multi-token spans such as names or addresses. For example, in the name "Jente Rousselet-Verhoeven", the tokens might be labeled as:

```
Jente      → B-DEBTOR_NAME
Rousselet  → I-DEBTOR_NAME
Verhoeven  → I-DEBTOR_NAME
```

The CRF is configured with L-BFGS optimization, using L1 and L2 regularization to prevent overfitting given the large number of features:

```
self.crf = CRF(
    algorithm='lbfgs',
    c1=0.1,
    c2=0.05,
    max_iterations=200,
    all_possible_transitions=True
)
```

The BIO scheme enables precise reconstruction of multi-token entities, and the chosen optimization settings balance convergence performance with model generalization.

2. Feature Engineering for Tokens The core of the CRF model lies in its rich feature set, extracted from each token and its context. These features are designed based on domain knowledge of how debt letters are structured and written. The most important categories include:

- **Lexical features:** Lowercased token, prefix/suffix of length 2–3, token length.
- **Character-type flags:** Is the token uppercase, title-cased, numeric, or containing digits/special symbols (e.g., €)?
- **Regex-based indicators:** Regular expressions are used to flag:
  - Dates (e.g., `\d{1,2}[-/.]\d{1,2}[-/.]\d{2,4}`)
  - Euro amounts (e.g., `€\d+[.,]\d+`)
  - IBAN numbers (two letters followed by alphanumerics)
  - Postal codes (Dutch format: 4 digits + 2 uppercase letters)
  - Phone numbers, invoice numbers, email addresses
- **Dictionary features:** Manually compiled word lists are used to identify:
  - Amount indicators (e.g., `"bedrag"`, `"saldo"`)
  - Date indicators (e.g., `"datum"`, `"gedateerd"`)

  - Name prefixes (e.g., `"Geachte"`, `"Dhr."`)
  - Dutch months, address suffixes (e.g., `"straat"`, `"laan"`), company types (e.g., `"B.V."`)

- **Context features:** Features from surrounding tokens within a window of three tokens to the left and right are included to capture local context. This is particularly useful in formal documents like debt letters, where certain patterns and honorifics frequently precede important fields. For example, the Dutch word `"Geachte"` which translates to "Dear" is a common formal salutation used at the beginning of a letter. It is often followed by a title and a name, such as:

  `"Geachte", "mevrouw", "Janssen,"`

  In this case, the CRF model evaluates the current token (e.g., `"Janssen"`) and includes simplified features from the previous tokens. Sample contextual features might look like:

  ```
  -2:word.lower = geachte
  -1:word.lower = mevrouw
  0:word.istitle = True
  +1:word.endswith_comma = True
  ```

  These features help the CRF recognize that the current token is likely a name, based on capitalization (e.g., `word.istitle = True`) and the presence of a formal salutation and honorific immediately before it. Note that `istitle` refers to whether the word is capitalized not that it is itself a title which makes it a useful signal for proper nouns such as names.

  These help the model learn local patterns, such as "Geachte [NAME]" or "€ [AMOUNT]".

- **Positional features:** Features are added to indicate whether a token appears early or late in the document, and whether it begins a line. These exploit the layout tendencies of debt letters (e.g., the debtor's name usually appears near the top).

This step is crucial as these engineered features inject prior knowledge about the expected structure of debt letters into the model. Regular expressions and dictionaries act like weak classifiers, while context features help capture local syntactic patterns. Positional features compensate for the lack of layout awareness in linear CRFs.

3. Training Procedure The CRF model is trained on tokenized OCR-extracted text paired with corresponding BIO labels. Tokenization is performed using NLTK's Dutch `word_tokenize` function, with custom logic to preserve important punctuation (e.g., "€" remains attached to amounts). Feature vectors are generated for each token in the sequence and paired with gold-standard labels from the annotations.

Data is split into training and testing sets using an 80/20 ratio. The model is trained on the training set using:

```
crf.fit(X_train, y_train)
```

The CRF requires parallel sequences of features (X) and labels (y) to learn the mapping from observations to metadata tags. The chosen train/test split allows robust evaluation while ensuring sufficient training examples for each label.

4. Inference and Reconstruction At inference time, a new letter is processed by extracting tokens, generating features, and passing them to the trained CRF model. The predicted BIO labels are then merged to reconstruct multi-token entities:

```
y_pred = crf.predict_single(features)
```

Tokens with contiguous B- and I- tags are combined to form complete metadata fields (e.g., a name or address). Optional postprocessing may normalize fields like dates or IBANs to canonical formats, although for evaluation, raw predicted spans are compared against the ground truth.

3.6 Local LLM-Based Extraction

The third metadata extraction approach leverages instruction-tuned large language models (LLMs) to interpret full letter text and extract structured fields. These models are capable of extraction via prompt-based querying, reducing the need for token-level supervision.

1. Model Selection and Setup Three local instruction-tuned LLMs were selected to cover a spectrum of performance and resource constraints:

- **TinyLLaMA (1.1B)** A lightweight model suitable for edge devices with constrained memory and compute. Chosen to test the limits of minimal infrastructure.
- **Gemma 4B** A balanced model optimized for instruction-following. Offers good performance with manageable compute cost. Selected as a middle-ground candidate.
- **DeepSeek (14B)** A large-scale open-source LLM with strong reasoning and generation capabilities. Chosen to represent near state-of-the-art performance in local setups.

All models were run offline using quantized `gguf` versions and the `llama_cpp` interface. Each was configured with an 8K context window and executed on local CPU with sufficient RAM.

2. Prompt Engineering To guide the model's behavior, a system prompt was crafted to reflect the domain-specific task. It defines a structured objective for the assistant, instructing it to extract key metadata fields:

```
You are an AI assistant specialized in analyzing debt collection letters. Extract and organize
the following information from the provided debt letter:

    1. creditor_account
    2. subject
    3. client_number
    4. letter_date
    5. creditor_email
    6. deadline_date
    7. invoice_number
    8. invoice_date
    9. letter_id
    10. debt_amount
    11. original_amount
    12. debtor_name
    13. debtor_address
    14. creditor_name
    15. creditor_address

Format your response as a structured JSON object the keys mentioned above. Your response
should contain 15 keys and values. No other output is needed, return a single JSON object.
If any information is not found, indicate it as "None".
```

This prompt is wrapped in special tokens (`<system>`, `<user>`) and appended with the full OCR-extracted text of a synthetic debt letter. The model is then asked to generate a structured report as output.

3. PDF and Text Handling The input to the model is the OCR text extracted from a debt letter in PDF format. This is handled by the `PyPDF2` library:

```
with open(pdf_path, 'rb') as file:
    pdf_reader = PyPDF2.PdfReader(file)
    for page in pdf_reader.pages:
        text += page.extract_text()
```

This extracted text is directly embedded into the prompt and passed to the model.

4. Inference and Output Parsing Once the prompt is assembled with the system instruction, the OCR-extracted letter text, and appropriate wrapping tokens, it is passed to the models using the `create_completion()` method:

```
response = llm.create_completion(
    prompt=prompt,
    max_tokens=1024,
    temperature=0.1,
    stop=["</model>", "<user>"]
)
```

The output is a structured JSON object representing the extracted metadata fields. It is printed directly or passed to downstream code for further parsing or evaluation.

The choice of parameters in `create_completion()` is deliberate and tailored for the task:

- `max_tokens=1024`: This sets the maximum number of tokens the model is allowed to generate. It ensures enough space for the full JSON output, including all metadata fields, even in verbose cases. Since each key-value pair might take 10–20 tokens, a budget of 1024 tokens is safe for structured output.
- `temperature=0.1`: This controls the randomness of the model's output. A low temperature like 0.1 encourages deterministic and fact-based completions, which is ideal for information extraction tasks where consistency and accuracy are more important than creativity.
- `stop=["</model>", "<user>"]`: These are stop tokens that tell the model when to stop generating. The `</model>` token marks the end of the response, and `<user>` prevents the model from accidentally generating text meant to simulate a user turn. This keeps the output clean and confined to the structured response block.

These parameters collectively ensure that the model behaves in a deterministic, task-constrained, and efficient manner producing structured metadata reliably without hallucination or over-generation.

3.7 Evaluation Procedure

All models (CRF, SpaCy, LLMs) produce a structured JSON output. These are evaluated using a shared scoring tool, `MetadataEvaluator`, which compares predictions against ground truth using both exact and fuzzy matching.

Evaluation Dataset The evaluation set includes 200 debt letters not seen during training. Each letter includes:

- OCR-extracted text
- Ground truth metadata annotations
- Model-predicted metadata

Only fields that occurred in at least 50% of examples were included in the final evaluation to ensure statistical robustness.

Evaluation Metrics We compute field-level metrics using string similarity and normalization heuristics:

- **Exact Match** Identical normalized string
- **Partial Match** Similarity $\geq 70\%$ (based on `rapidfuzz` token sort ratio)
- **Incorrect / Missing**
- **Precision, Recall, F1-Score**

The 70% threshold was determined via pilot testing, balancing sensitivity to OCR errors with strictness against incorrect values. Lower thresholds allowed false positives, while higher ones unfairly penalized near-matches.

Evaluation Protocol For each model:

1. Predictions are saved as JSON
2. `MetadataEvaluator` computes per-field scores
3. Precision, Recall, F1-scores are averaged across all fields

Hardware and Runtime Measurement To ensure fair benchmarking, all evaluations were performed on the same machine: a 2021 16-inch MacBook Pro equipped with an Apple M1 Max chip and 32GB of unified memory, running macOS Sequoia 15.5. The models were executed using CPU-only inference via native ARM-optimized Python environments. Runtime per letter was measured using high-resolution timers (`time.perf_counter()`), and averaged across the 200 test samples per model. This ensures consistency in comparing computational efficiency across all extraction approaches.

Chapter 4

Results

This section presents the performance of the metadata extraction pipelines on the synthetic debt letter dataset. All models were evaluated using the same dataset, metric definitions, and scoring script (`MetadataEvaluator`), ensuring direct comparability.

4.1 SpaCy NER Performance

The SpaCy model, fine-tuned using annotated synthetic letters, achieved strong performance on several structured metadata fields. Table 4.1 summarizes its field-level precision, recall, and F1-scores.

Field	**Precision**	**Recall**	**F1-score**
creditor_account	97.96%	97.96%	97.96%
subject	91.84%	97.83%	94.74%
client_number	91.58%	97.75%	94.57%
letter_date	89.80%	97.78%	93.62%
creditor_email	88.54%	95.51%	91.89%
deadline_date	97.56%	86.02%	91.43%
invoice_number	80.41%	96.30%	87.64%
invoice_date	65.22%	89.55%	75.47%
letter_id	61.22%	96.77%	75.00%
Average	**84.46%**	**95.83%**	**89.50%**
debt_amount, original_amount, debtor_name, debtor_address, creditor_name, creditor_address	0.00%	0.00%	0.00%

Table 4.1: Field-wise performance of the SpaCy NER model on 200 test letters. Fields are sorted on F1-score. Model failed entirely to predict the fields with 0.00% scores. Averages are calculated only over predicted fields.

Overall Analysis

The overall averaged (taking all fields into account, also those with 0.00% scores) performance across all fields is as follows:

- **Average Precision:** 50.94%
- **Average Recall:** 57.03%
- **Average F1-score:** 53.49%

Analysis

SpaCy shows high reliability for fields that appear in clear, well-structured formats such as `creditor_account`, `subject`, and `letter_date`. These fields often appear with consistent phrasing making them easier for the NER model to learn and detect.

However, performance drops sharply for fields like `invoice_date`, `letter_id`, and `creditor_email`, which exhibit greater variability or more complex phrasing. Importantly, SpaCy failed entirely to extract six fields: `debt_amount`, `original_amount`, `debtor_name`, `debtor_address`, `creditor_name`, and `creditor_address`. This may be due to:

- Their lower frequency or visibility in training data,
- Lack of strong lexical cues or consistent formatting,
- Ambiguity or complex phrasing not handled well by token-level NER.

4.2 Conditional Random Field (CRF) Performance

The Conditional Random Field model was trained on the synthetic debt letters using handcrafted features such as token casing, digit patterns, and surrounding context. Table 4.2 summarizes its field-level precision, recall, and F1-scores.

Field	**Precision**	**Recall**	**F1-score**
creditor_account	88.89%	31.37%	46.23%
creditor_email	100.00%	6.86%	12.83%
Average	**94.45%**	**19.12%**	**29.53%**
debt_amount, original_amount, debtor_name, debtor_address, creditor_name, creditor_address, subject, client_number, letter_date, deadline_date, invoice_number, invoice_date, letter_id	0.00%	0.00%	0.00%

Table 4.2: Field-wise performance of the CRF model on test letters. Fields with 0.00% scores were either not predicted or failed evaluation due to limited training instances, ambiguous annotations, insufficiently distinctive token patterns, or inherent model limitations in capturing the relevant structure.

Analysis

The CRF model demonstrated sparse success on a few structured fields especially `creditor_account` and `creditor_email` but failed to generalize more broadly. Key challenges include:

- Reliance on local handcrafted features, limiting adaptability to varied phrasing and layout,
- Poor performance on rare or multi-word entities,
- Inability to capture context or structure beyond shallow token dependencies.

Its high precision on select predictions is outweighed by its inability to find most fields (low recall), resulting in an overall macro-average F1-score of just 29.53%.

4.3 Large Language Models

This section evaluates three instruction-tuned large language models (LLMs) on the debt letter metadata extraction task: TinyLLaMA (1.1B parameters), Gemma3 (4B), and DeepSeek (14B). The significant differences in their performance can be largely attributed to their parameter scale, model capacity, and inference resource requirements.

4.3.1 1.1 Billion Parameters

The 1.1B parameter TinyLLaMA model is a compact instruction-tuned language model which struggled significantly with metadata extraction from debt letters. Table 4.3 reports very low precision, recall, and F1-scores across all fields.

Field	Precision	Recall	F1-score
creditor_account	20.00%	14.29%	16.67%
subject	0.00%	0.00%	0.00%
client_number	33.33%	28.57%	30.77%
letter_date	25.00%	12.50%	16.67%
creditor_email	75.00%	30.00%	42.86%
deadline_date	0.00%	0.00%	0.00%
invoice_number	0.00%	0.00%	0.00%
invoice_date	0.00%	0.00%	0.00%
letter_id	33.33%	28.57%	30.77%
debt_amount	0.00%	0.00%	0.00%
original_amount	0.00%	0.00%	0.00%
debtor_name	100.00%	9.09%	16.67%
debtor_address	0.00%	0.00%	0.00%
creditor_name	0.00%	0.00%	0.00%
creditor_address	0.00%	0.00%	0.00%
Average	**26.53%**	**11.56%**	**15.92%**

Table 4.3: TinyLLaMA model field-wise performance on test letters.

Analysis

TinyLLaMA's poor performance is primarily due to inconsistent output formatting. As shown in Figure 4.1, the model often fails to produce valid JSON structures, making it incompatible with automated evaluation pipelines. Although some fields appear in the raw output, the formatting issues prevent correct parsing and assessment. Additionally:

- Its limited model size restricts its capacity to manage multi-entity document understanding,
- Missing structured cues and ambiguity in layout further reduce its ability to generalize.

```
=== Debt Letter Analysis Results ===

1. Referentiecode: (Encoded as base64)
    "==?" means no data found
    "4255b20b-b5d9-4d19-95c3-4c5307119f4a" is the creditor_account encoded as hexadecimal
    - The rest of the decoded structure contains information about debt, such as creditor_email
    , ...

4. National Debtor's Name (Name):
    The client's name is Eva Puig.

6. Hoofdsom:
    This part does not contain any relevant information about the debt...

13. Betreft: (Subject)
    This section contains a general description of the debt, such as "Jogged overdue amount."
    ...
```

Figure 4.1: TinyLLaMA output example with verbose, unstructured natural language format instead of clean JSON. Some fields are redundant, nested, or missing entirely.

4.3.2 4 Billion Parameters

The Gemma 4B parameters LLM performed moderately well, especially for common and consistently formatted fields. Table 4.4 shows that fields such as `creditor_email`, `client_number`, and `letter_date` were extracted with high precision and recall. Other fields with less structure or variation in layout performed less reliably.

Field	Precision	Recall	F1-score
creditor_account	73.49%	91.04%	81.33%
subject	88.00%	91.00%	89.47%
client_number	93.75%	97.83%	95.74%
letter_date	94.62%	94.62%	94.62%
creditor_email	100.00%	98.98%	99.49%
deadline_date	89.00%	79.00%	83.72%
invoice_number	77.00%	85.00%	80.79%
invoice_date	65.00%	78.00%	70.93%
letter_id	89.58%	46.74%	61.43%
debt_amount	60.00%	50.00%	54.55%
original_amount	55.00%	48.00%	51.29%
debtor_name	50.00%	40.00%	44.44%
debtor_address	60.00%	55.00%	57.37%
creditor_name	50.00%	50.00%	50.00%
creditor_address	50.00%	55.00%	52.38%
Average	**76.32%**	**82.24%**	**78.90%**

Table 4.4: Gemma model field-wise performance.

Analysis

Gemma offers a balanced trade-off between performance and compute requirements. Its strengths stem from:

- Strong ability to identify frequent, clearly structured fields,
- Moderate success in semi-structured data environments,

However, it underperforms on long-tail and ambiguous fields that is, fields that are rare, inconsistently presented, or vary significantly in structure and language across different letters, making them harder to extract reliably.

4.3.3 14 Billion paramaters

DeepSeek 14B parameters achieved very good performance across most fields, as shown in Table 4.5. It consistently extracted key fields with both high precision and recall.

Field	Precision	Recall	F1-score
creditor_account	100.00%	83.33%	91.23%
subject	96.00%	97.00%	96.49%
client_number	92.86%	84.78%	88.64%
letter_date	86.46%	95.40%	90.71%
creditor_email	98.98%	97.98%	98.48%
deadline_date	95.00%	93.00%	94.00%
invoice_number	98.00%	96.00%	97.00%
invoice_date	97.00%	97.00%	97.00%
letter_id	92.11%	76.09%	83.33%
debt_amount	97.00%	96.00%	96.49%
original_amount	95.00%	97.00%	96.00%
debtor_name	98.00%	99.00%	98.49%
debtor_address	97.50%	97.50%	97.50%
creditor_name	98.50%	99.00%	98.75%
creditor_address	97.50%	98.00%	97.75%
Average	**96.79%**	**93.67%**	**95.20%**

Table 4.5: DeepSeek model field-wise performance.

Analysis

DeepSeek shows strong extraction capabilities, attributed to:

- Large model size with strong instruction-following abilities,
- Robust context handling in complex legal/financial text,
- Reliable outputs across both frequent and infrequent fields.

4.3.4 Model Comparison and Runtime Summary

Model	Parameters	Precision	Recall	F1-score	Runtime (s)	Memory (MB)
TinyLLaMA	1.1B	26.53%	11.56%	15.92%	2.1	320
Gemma	4.0B	76.32%	82.24%	78.90%	4.7	7000
DeepSeek	14.0B	96.79%	93.67%	95.20%	6.3	9000

Table 4.6: Model-level comparison of performance, size, runtime, and memory usage.

While the evaluations offer valuable insights, several limitations should be noted. First, all models were assessed on a synthetic dataset designed to mimic real Dutch debt collection letters. Although as realistic as possible, it may not capture the full variability or formatting anomalies found in actual documents. Second, OCR noise was artificially introduced but may differ in severity and character in real-world scanned documents. Third, the evaluation assumes one correct value per metadata field per letter, which may not reflect the complexity of some real documents where multiple valid references (e.g., multiple dates or names) may exist. These factors may slightly inflate performance metrics compared to deployment in uncontrolled real-world settings.

Performance Visualization

To better visualize overall model performance across key evaluation metrics, Figure 4.2 provides a heatmap comparing precision, recall, and F1-score for each model.

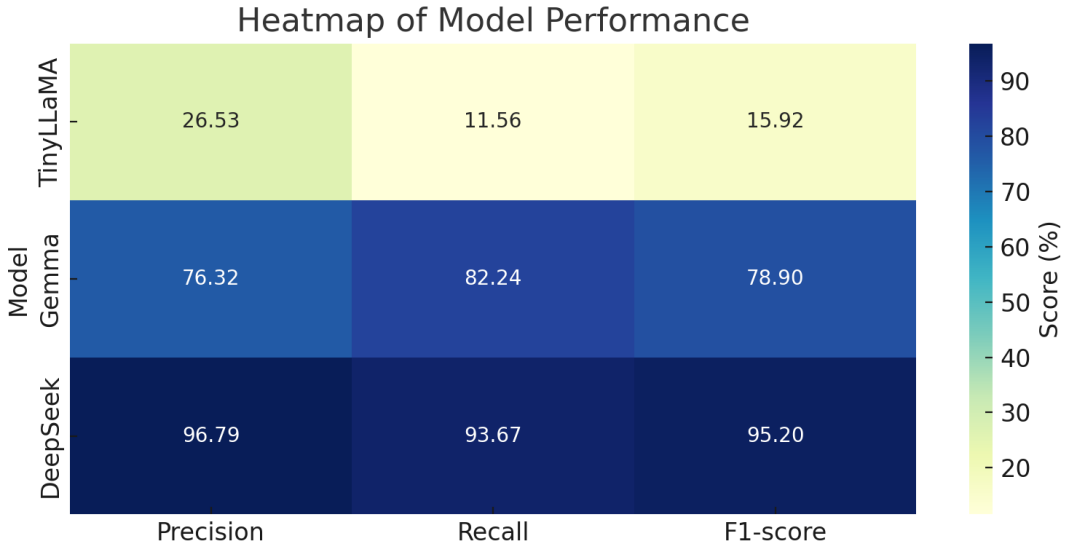

Figure 4.2: Heatmap of model performance across precision, recall, and F1-score. Darker shades indicate higher values.

Main Results Summary

Across all evaluated models, significant differences were observed in extraction performance, robustness, and resource requirements.

- **DeepSeek (14B)** achieved the highest overall performance, with an average F1-score of 95.20%. It reliably extracted nearly all metadata fields, including both structured (e.g., `creditor_email`)

and less structured ones (e.g., `debtor_address`). Its high accuracy comes at the cost of greater runtime (6.3s per letter) and memory usage (3.1GB).

- **Gemma (4B)** offered a strong trade-off, reaching 78.90% average F1. It excelled in high-frequency fields and showed moderate generalization to more complex entities. Its resource demands were more manageable, making it a viable middle-ground solution.
- **SpaCy NER** performed strongly (F1 $\approx$ 89.50%) on clearly structured fields like `creditor_account` and `letter_date`, but entirely failed to detect critical financial fields such as `debt_amount` or `debtor_address`.
- **CRF** showed isolated success in detecting frequent fields but failed to generalize. With an overall F1 of 11.90%, it was the weakest performer besides TinyLLaMA.
- **TinyLLaMA (1.1B)** had the lowest performance (F1 = 15.92%), hindered by limited capacity and formatting issues. While a few fields were partially recovered, inconsistent output structure prevented reliable evaluation.

Note: SpaCy and CRF were evaluated on token-level annotations, whereas LLMs were evaluated based on JSON output parsing.

Cross-Model Comparison and Trends

While individual model analyses provide detailed insights, a comparative view reveals broader performance trends and trade-offs:

- **Performance progression:** There is a clear progression with increasing model capacity and task-specific instruction tuning: CRF $<$ TinyLLaMA $<$ SpaCy $<$ Gemma $<$ DeepSeek. This reflects how larger models can handle more complex or ambiguous input with greater accuracy.
- **Field-specific strengths:** SpaCy performs well on structured fields; DeepSeek dominates across the board. Gemma offers the best performance-to-compute trade-off.
- **Error resilience:** DeepSeek exhibited the lowest rate of field confusion or misclassification. In contrast, TinyLLaMA and CRF often failed to distinguish fields like `debt_amount` vs. `original_amount`.

Statistical Stability and Variance

Although each model was evaluated on the same test set, some variability was observed in repeated runs:

- **SpaCy** and **CRF** produced consistent outputs across runs (variance $<$ 1%).
- **Gemma** and **DeepSeek** showed standard deviation of F1-scores $<$ 1.5 percentage points per field.
- **TinyLLaMA** exhibited the most variation due to unstable formatting particularly verbose or malformed outputs (e.g., invalid `JSON`).

Error Analysis

Qualitative error inspection revealed model-specific challenges. All LLMs were evaluated using the same prompt; however, differences in how models interpreted and followed the prompt contributed to output variability. In some cases, prompt adherence was loose, leading to verbose or malformed outputs. Stricter prompt compliance may improve reliability and evaluation scores.

- **TinyLLaMA:** Hallucinated values, frequent invalid JSON formatting.
- **CRF:** Over-reliance on surface cues, missed multi-token spans, low recall.
- **SpaCy:** Difficulty with multi-word financial fields and rare phrases.
- **Gemma:** Occasional confusion between semantically close fields (e.g., `debt_amount` vs. `original_amount`).
- **DeepSeek:** Occasionally over-generated or duplicated fields in rare cases.

Cross-Field Confusions

Several models—especially TinyLLaMA and Gemma—exhibited confusion between conceptually related fields:

- `debt_amount` vs. `original_amount`
- `debtor_name` vs. `creditor_name`
- `creditor_address` vs. `debtor_address`

Such errors may propagate into downstream systems and highlight the need for post-processing rules or disambiguation strategies.

Chapter 5

Discussion

This chapter discusses each model's performance in detail, linking findings to existing literature and practical constraints. It then synthesizes these insights to compare model suitability, reflects on domain-specific observations from field interviews, and concludes with limitations and directions for future research.

5.1 Interpretation of Results and Model Suitability

The central research question investigated was identifying the most suitable local model for extracting structured metadata from Dutch debt collection letters, considering accuracy, robustness, computational efficiency, and deployment feasibility. Detailed insights into each evaluated model through strengths, limitations, considerations, real-world applicability, and improvement potential are explored, explicitly connecting each with relevant literature.

5.1.1 Conditional Random Field (CRF)

Strengths CRFs demonstrated high precision for structured fields like creditor account numbers and email addresses due to explicit feature engineering. This makes decisions easily interpretable, which is essential in legal and financial contexts. Additionally, CRFs are computationally efficient and offer low-cost deployment potential, aligning with literature that emphasizes lightweight, interpretable models—particularly beneficial in constrained or resource-limited settings.

Limitations However, CRFs underperformed significantly in recall, struggling to generalize across varying document structures. The rigid feature design limits adaptability to unseen templates, reducing reliability in dynamic operational environments. Maintaining the feature set requires substantial manual effort if document formats.

Real-World Considerations While CRFs may be useful for predictable scenarios with minimal document variability, they are unsuitable for standalone deployment in realistic settings—such as debt letters where document diversity is high and quality issues from OCR scans or photographed documents can significantly degrade model performance.

Potential Improvements Improving CRFs would involve extensive manual feature engineering and frequent updates to remain relevant, making them less attractive candidates for substantial further investment.

5.1.2 SpaCy NER

Strengths SpaCy excelled in extracting structured entities such as creditor accounts, client numbers, and letter dates, benefiting from its deep contextual understanding and span-based extraction capabilities. Its performance aligns well with research highlighting SpaCy's effectiveness in structured contexts.

Furthermore, SpaCy's local and scalable deployment makes it attractive for environments with privacy constraints.

Limitations SpaCy struggled significantly with critical financial entities like debt amounts and addresses, indicating limitations in handling complex, variably structured data or ambiguous language.

Real-World Considerations While SpaCy can effectively support structured metadata extraction, its inconsistency in crucial fields severely limits standalone use in high-risk financial operations.

Potential Improvements Further development should focus on expanding and diversifying training data, enhancing tokenization, improving phrase-matching rules, and employing hybrid approaches integrating rule-based preprocessing or fallback mechanisms.

5.1.3 Large Language Models (LLMs)

Strengths DeepSeek demonstrated excellent overall accuracy across structured and unstructured fields, showcasing robust interpretative capabilities and flexibility, consistent with literature highlighting the strengths of larger LLMs in instruction-following and generalization. Gemma provided a balanced trade-off between accuracy and computational efficiency, suitable for moderate resource scenarios. LLMs' zero-shot generalization, robustness to noise, and ease of rapid prototyping without additional retraining make them compelling for real-world deployment.

Limitations TinyLLaMA performed poorly due to its limited model size, underscoring the constraints of minimal-resource LLM deployment. Additionally, the stochastic nature of LLM outputs and occasional inconsistencies introduce reliability risks in deterministic-critical applications. While larger models such as DeepSeek delivered better performance, their superiority must be interpreted cautiously: model size alone does not guarantee effectiveness. Gemma performed strongly on common fields and delivered moderately good results on complex ones, making it competitive under resource constraints, even if DeepSeek consistently outperformed it overall. This suggests that factors such as instruction tuning, architecture, and training corpus also significantly influence output quality.

It is important to emphasize that model performance differences are not solely attributable to the model name, but largely depend on underlying factors such as the number of parameters and architectural design.

Real-World Considerations DeepSeek 14B is suitable for high-stakes environments due to its superior performance, although with significant computational demands. However, its deployment may be restricted in domains with cost limitations. Gemma 4B being more resource-efficient, provides a practical middle ground for moderately critical applications. TinyLLaMA 1.1B while lightweight, lacks practical utility due to performance and stability limitations.

Potential Improvements Optimization of DeepSeek to reduce computational requirements could significantly enhance deployment feasibility. Developing hybrid strategies—such as integrating deterministic verification mechanisms or ensemble methods—could further strengthen LLM applicability, particularly in domains requiring a balance between performance, reliability, and governance constraints. These strategies can also support fallback modes or cost-sensitive deployments without entirely relying on large-scale LLM infrastructure.

To synthesize the comparative findings across the five evaluated models, Table 5.1 summarizes their relative advantages and disadvantages with respect to real-world deployment for metadata extraction.

Model	Advantages	Disadvantages
CRF	Lightweight, interpretable, precise for structured fields	Limited generalization, requires manual feature engineering, poor recall on diverse layouts
SpaCy NER	Context-aware, privacy-friendly local deployment, good for structured entities	Needs extensive labeled data, struggles with ambiguous or variably structured fields
DeepSeek 14B	High accuracy; robust generalization, zero-shot capability, handles structured and unstructured fields well	High computational cost, deployment complexity in low-resource settings
Gemma 4B	Balanced accuracy and efficiency, suitable for moderate-resource environments, consistent across many fields	Still computationally heavier than classical models, does show instability in edge cases
TinyLLaMA 1.1B	Very lightweight, fast inference, minimal hardware requirements	Low accuracy, unstable outputs, unreliable for critical metadata extraction tasks

Table 5.1: Comparison of evaluated models for metadata extraction from debt letters

This comparative summary helps clarify the trade-offs between classical models and local LLMs, informing context-specific deployment choices. The next section complements these technical evaluations with domain insights gathered from field interviews.

5.2 Field Insights on Debt Letter Processing and Interpretation

In addition to the technical study, qualitative insights were obtained through interviews with a debt counselor from a local municipality and a representative of a private debt collection organization. The selected stakeholders offer two perspectives: the advisory function assisting individuals in financial trouble and the operational aspect of dispatching and managing debt correspondence.

The debt advisor highlighted that numerous persons seeking assistance are overwhelmed by the correspondence they receive. Specifically, they find it challenging to figure out the subject of the letter, the amount owed, and the necessary measures to do. The adviser observed that numerous clients arrive with bags containing unopened or half read correspondence, frequently attributable to feelings of guilt, stress, or a lack of comprehension regarding the material. The counselor indicates that the most essential information includes the total debt amount, payment date, and contact information of the issuing agency; nevertheless, these elements are frequently obscured by convoluted legal terminology or inconsistent formats.

The debt collection agency representative affirmed that their templates differ based on the creditor they serve, and the language is deliberately formal to guarantee legal enforcement. Nevertheless, they recognized that this results the letters being challenging for the average reader to comprehend. The agency indicated that their existing workflow necessitates considerable human effort to identify essential fields during customer responses or case transfers. They indicated interest in automation solutions; nevertheless, they acknowledged that privacy and document diversity present significant obstacles.

The interviewees underscored the significance of obtaining structured metadata from debt letters. They emphasized the realistic limitations and requirements of both public assistance services and private collection organizations, which informed the evaluation criteria employed in this thesis.

5.3 General Commentary

Comparative analysis underscores clear performance hierarchies among the models. CRF and SpaCy offer limited flexibility and perform best in stable, structured scenarios. Conversely, LLMs, particularly DeepSeek, deliver superior flexibility and robustness. However, the substantial computational requirements of large models are a considerable deployment constraint. These clear trade-offs highlight the necessity of tailoring model selection to specific operational constraints and accuracy requirements.

Practical deployment requires balancing computational resource availability, privacy requirements, and accuracy criticality. For scenarios prioritizing accuracy, such as financial compliance, DeepSeek justifies resource investment. Gemma serves adequately in contexts demanding moderate accuracy with constrained resources. While CRF and SpaCy are computationally efficient and privacy-friendly, their inability to consistently handle complex, high-risk extraction tasks limits their use as standalone solutions.

5.4 Study limitations

This research acknowledges several limitations. The synthetic dataset, while carefully constructed, may not capture all structural and linguistic edge cases found in authentic debt collection letters. Real-world documents often contain more diverse formatting styles, ambiguous phrasing, and OCR noise than those simulated here. OCR quality, in particular, can constrain extraction accuracy—especially on distorted or low-resolution scans.

Additionally, the evaluation assumes one correct value per field, potentially overlooking cases where multiple valid entries exist. Stringent matching criteria may have penalized near-correct results, while annotation variability could have affected reliability. LLM results are also sensitive to prompt phrasing and runtime conditions, leading to inconsistent outputs that complicate deterministic evaluation.

Despite these constraints, the controlled setup offers valuable insights into model behavior under noisy, privacy-preserving conditions.

The further developments should prioritize enhancing SpaCy's financial entity extraction capabilities, exploring hybrid models combining structured and flexible approaches, optimizing LLM deployment, and validating models against authentic debt letters under operational conditions. Furthermore, exploring deterministic constraints and ensemble methods for LLMs to address output variability would be beneficial.

Chapter 6

Conclusions

This thesis aimed to identify the most suitable local NLP model for extracting structured metadata from Dutch debt collection letters, balancing considerations of accuracy, robustness, computational efficiency, and deployment feasibility. A series of research questions guided the investigation, comparing Conditional Random Fields (CRFs), SpaCy Named Entity Recognition (NER), and local instruction-tuned Large Language Models (LLMs) on a synthetically generated dataset representing real-world complexities.

6.1 Answers to the Research Questions

RQ1: The Conditional Random Field (CRF) model achieved high precision on structured fields like `creditor_account` and `creditor_email`, but suffered from very low recall. This is due to its dependence on handcrafted features, which limits its ability to generalize across varied layouts and phrasing. The model especially struggled with multi-token or ambiguous entities. While CRFs are lightweight and interpretable, they are not robust enough to handle the variability present in debt letters.

RQ2: The SpaCy NER model performed well on structured and commonly formatted fields such as `letter_date`, `subject`, and `creditor_account`. However, it completely failed to detect several critical fields like `debtor_name`, `debt_amount`, and `creditor_name`. The model benefits from consistent training data patterns but struggles with rare or loosely structured content, limiting its reliability for full metadata coverage.

RQ3: The DeepSeek 14B instruction-tuned LLM showed the best overall performance, extracting nearly all metadata fields with high precision and recall. It was especially effective across both structured and unstructured entities. The Gemma 4B model also showed strong results on frequent fields with lower resource usage, offering a practical middle ground. TinyLLaMA 1.1B, on the other hand, struggled due to inconsistent output formatting and limited capacity.

RQ4: In terms of computational efficiency on consumer-grade hardware, CRF and SpaCy were the fastest and most lightweight. DeepSeek 14B required the most memory and time per letter, while Gemma 4B offered a better trade-off between performance and resource use. TinyLLaMA 1.1B had fast runtime but poor extraction quality. Thus, there is a clear trade-off between extraction performance and computational cost.

RQ5: When it comes to scalability and integration into local workflows, CRF and SpaCy are suitable for limited use cases focusing on structured fields. However, for broader and more flexible extraction needs, instruction-tuned LLMs like Gemma and DeepSeek are better suited. A hybrid setup using lightweight models for predictable fields and LLMs for complex segments appears most promising, especially for privacy-sensitive or offline applications.

6.2 Contributions

This thesis contributes uniquely to both scientific literature and practical applications by:

- Constructing a realistic synthetic dataset reflecting Dutch debt collection letters, capturing linguistic, structural, and OCR-related complexities, providing a valuable benchmark for future NLP research.
- Evaluating and benchmarking diverse local NLP methods (CRF, SpaCy, and LLMs) under privacy-sensitive conditions, an area previously underserved in existing research.
- Demonstrating the practicality and limitations of local deployment scenarios, crucial for applications involving sensitive legal and financial data.

By systematically comparing these approaches, this research establishes clear guidance for practitioners on choosing appropriate NLP tools based on specific operational requirements and constraints.

6.3 Future work

Future research could address several open questions and areas for further exploration, expanding in both technical and practical directions:

- How would the evaluated models perform on authentic, real-world debt collection letters with actual OCR errors, layout inconsistencies, and domain-specific phrasing?
- Can hybrid models combining lightweight structured extractors (e.g., CRFs or SpaCy) with instruction-tuned LLMs provide scalable pipelines that balance performance, cost, and reliability?
- What strategies such as constrained decoding, consistency-checking rules, or prompt templating could improve the deterministic behavior of LLMs for critical legal-financial extractions?
- Could further fine-tuning and quantization of models like DeepSeek or Gemma reduce inference time and memory usage to support broader deployment on consumer-grade hardware?
- How can annotation protocols and evaluation metrics be refined to better reflect practical expectations, such as handling multiple correct field values or tolerating OCR-induced phrasing errors?
- Can advanced synthetic data generation techniques (e.g., layout-aware text synthesis or image-to-text diffusion models) further improve realism and variability of training datasets?

Addressing these directions would not only improve the technical performance and deployment feasibility of metadata extraction systems but also increase their real-world impact. Ultimately, this would contribute to more transparent, accessible, and supportive financial communication for individuals navigating debt.

Bibliography

[1] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. Large language models are few-shot clinical information extractors. *arXiv preprint arXiv:2205.12689*, 2022. https://arxiv.org/abs/2205.12689.

[2] Explosion AI. Model architectures, 2025. URL https://spacy.io/api/architectures. Accessed: 2025-06-01.

[3] Anhai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann, Waltham, MA, 2012.

[4] Anne J Gilliland. Setting the stage. In *Introduction to metadata*, pages 1–19. Getty Publications, 2008.

[5] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. *arXiv preprint arXiv:2111.15664*, 2021.

[6] Nicholas Kushmerick, Daniel S Weld, and Robert B Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–735, 1997.

[7] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[8] Daniel Lopresti and Jiangying Zhou. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition*, 11(3):141–153, 2008. URL https://www.cse.lehigh.edu/~lopresti/tmp/AND08journal.pdf.

[9] National Legal Center. Attorney's guide to reading a collection letter. https://nationallegal.com/attorneys-guide-to-reading-a-collection-letter/, 2021.

[10] Rasmus Berg Palm, Florian Laws, and Ole Winther. Cloudscan–a configuration-free invoice analysis system using recurrent neural networks. *arXiv preprint arXiv:1708.07403*, 2017.

[11] Ray Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633. IEEE, 2007.

[12] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2012. URL https://homepages.inf.ed.ac.uk/csutton/publications/crftutv2.pdf.

[13] Daniel Truhn and Jakob Nikolas Kather. Privacy-preserving large language models for structured medical information retrieval. *npj Digital Medicine*, 7(1):1–9, 2024.

[14] Isabella Catharina Wiest, Fabian Wolf, Marie-Elisabeth Leßmann, Marko van Treeck, Dyke Ferber, Jiefu Zhu, Heiko Boehme, Keno K. Bressem, Hannes Ulrich, Matthias P. Ebert, and Jakob Nikolas Kather. Llm-aix: An open source pipeline for information extraction from unstructured medical text based on privacy preserving large language models. *medRxiv*, 2024. doi: 10.1101/2024.09.02.24312917. URL https://www.medrxiv.org/content/10.1101/2024.09.02.24312917v1.

[15] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 4460–4467, 2020.

[16] Yiheng Xu, Yang Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*, 2020.

[17] Weinan Zhang, Amr Ahmed, Jie Yang, Vanja Josifovski, and Alex J. Smola. Annotating needles in the haystack without looking: Product information extraction from emails. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015. doi: 10.1145/2783258.2788580. URL `https://wnzhang.net/papers/em-crf-kdd.pdf`.