University of Hamburg

Bachelor Thesis

Prompt Engineering for Slot Filling using Large Language Models

Author:
Noah Kleiner
Matr.nr.:
7311722
Course of Study:
Informatics, B.Sc.

Supervisor:
Prof. Dr. Ricardo Usbeck
Secondary Supervisor:
Cedric Möller

A thesis submitted in fulfilment of the requirements for the degree of Bachelor of Science

in the

Semantic Systems Research Group
Department of Informatics

February 5, 2024

Abstract

The aim of this thesis is to investigate how large language models may be adopted for the purposes of extracting domain-specific structured information from medical report documents. Solving the structured information extraction problem at hand requires overcoming two major constraints, due to the visual structure of the given documents and the limited number of documents available for training purposes.

Firstly, this thesis therefore explores how a fine-tuned multi-modal language model may be employed to segment the textual contents of the documents, according to the constituent semantic sections within a medical report document. Secondly, this thesis further explores how framing the structured information extraction problem at hand as a slot filling task reduces the problem to a natural language comprehension challenge and enables the use of in-context examples.

Notably, this thesis proposes an approach for selecting suitable in-context examples ahead of inference time, based exclusively on the available ground truth labels. More specifically, this thesis introduces an algorithm for performing a semi-random greedy search over the possible combinations of in-context examples in two separate phases, concerning the rotation and minification of candidate examples respectively.

Finally, this thesis evaluates the implementation of the resulting domain-specific document analysis system, demonstrating a high overall recall with regards to the extraction of the desired structured information from medical report documents.

The complete source code for the work on this thesis including the implementation of the domain-specific document analysis system is available at `https://github.com/semantic-systems/medical-slot-filling`.

Zusammenfassung

Das Ziel der vorliegenden Arbeit ist es, zu untersuchen, wie große Sprachmodelle (engl. large language models) dazu verwendet werden können, domänenspezifische strukturierte Informationen aus Arztbriefen zu extrahieren. Um das betrachtete Problem zu lösen, ist es insbesondere erforderlich, zwei erschwerende Umstände zu überwinden, welche sich aus der visuellen Struktur der Arztbriefe, sowie der geringen Anzahl der zum Zweck des Trainings zur Verfügung stehenden Arztbriefe ergeben.

Die vorliegende Arbeit erforscht erstens, wie ein multimodales Sprachmodell durch das Training mit einem eigenen Datensatz dazu eingesetzt werden kann, die textuellen Inhalte der Arztbriefe gemäß der verschiedenen semantischen Abschnitte innerhalb der Dokumente zu segmentieren. Die vorliegende Arbeit erforscht zweitens, wie das betrachtete Problem durch die gezielte Modellierung als Slot-Filling-Aufgabe auf das Problem des Verstehens natürlicher Sprache reduziert werden kann, wodurch der Einsatz von Kontext-Beispielen (engl. in-context examples) ermöglicht wird.

Die vorliegende Arbeit schlägt diesbezüglich einen Ansatz vor, welcher die Auswahl von geeigneten Kontext-Beispielen bereits vor Inferenz-Zeit allein auf Basis der für die Kontext-Beispiele vorhandenen Ground-Truth-Labels ermöglicht. Die vorliegende Arbeit stellt weiterhin einen Algorithmus vor, welcher eine gierige Suche über die möglichen Kombinationen von Kontext-Beispielen in zwei Phasen implementiert, die jeweils die Rotation und Minimierung der möglichen Kontext-Beispiele betreffen.

Die vorliegende Arbeit führt abschließend eine Evaluation des für die Analyse von Arztbriefen implementierten Systems aus, um zu demonstrieren, dass insgesamt ein hoher Wert bezüglich der Recall-Metrik bei der Extraktion der domänenspezifischen strukturierten Informationen erreicht wird.

Der Quellcode für die vorliegende Arbeit, einschließlich des implementierten Systems für die domänenspezifische Analyse von Arztbriefen, kann unter `https://github.com/semantic-systems/medical-slot-filling` gefunden werden.

Contents

1 Introduction

1.1 Motivation

The University Medical Center Hamburg-Eppendorf (Universitätsklinikum Hamburg-Eppendorf, UKE) currently maintains a database containing various kinds of patient information collected from a large number of medical report documents as part of an ongoing research project into cases of acute heart failure and related conditions.

The database is currently updated through a process where relevant information mentioned in available medical report documents or collected in patient interviews is manually entered into the database. Notably, this task requires a certain base level of medical knowledge and is therefore usually performed by medical personnel.

Considering the high cost of human labor involved in this process, the UKE has approached the Semantic Systems research group at the University of Hamburg about the possibility of building an automated system capable of extracting the desired structured information from medical report documents, which are already available in digital form for all patients recently treated at the UKE, using artificial intelligence.

1.2 Problem Statement

We define the problem at hand as the challenge of extracting well-known types of structured information from medical report documents. The desired information must be extracted according to a target schema, which is derived from and closely aligned with the actual schema of the research database maintained at the UKE.

The target schema concretely consists of 93 distinct variables. For each of these well-known variables, a suitable value may be determined based on the textual contents of a given medical report document. The target schema is further divided into a number of different groups of related variables, such as pre-existing conditions or new diagnoses made during a specific hospitalization case of a patient at the UKE.

The value data type of each variable in the target schema is either boolean (where the value constitutes a binary classification), textual (where an exact value is extracted from the literal text of a medical report document), or of the enumeration type (where the value is selected from a pre-defined range of possible choices).

For example, the boolean variables "Arterial hypertension" and "Diabetes mellitus" both indicate the presence of certain pre-existing conditions. On the other hand, the textual variable "pH at admission" describes an exact measurement taken at the time the patient was admitted to the hospital during the standard admission process.

Notably, the majority of the variables in the target schema is of the boolean type (60 variables; 65%), while the second most common value data type is enumeration (19 variables; 20%), closely followed by the textual type (14 variables; 15%).

We are provided with corpus of 151 individual medical report documents available for the purposes of our work. The medical report documents in the corpus were created at the UKE as part of the standard patient treatment process. The medical report documents present in the corpus concern the treatment of cardiological conditions.

The UKE has collected patient consent and undertaken an effort to redact sensitive patient information for all the medical report documents present in the corpus.

We are further provided with the ground truth value of each variable in the target schema for every medical report document present in the corpus. The ground truth data has been manually compiled at the UKE by personnel with medical knowledge.

1.3 Research Questions

- **RQ1:** How can we segment input documents to overcome the limitations of LLMs with regards to the maximum number of input tokens?

One inherent limitation of large language models (LLMs) stems from the fact that the size of the context window is finite, meaning that they can only accept up to a builtin maximum number of input tokens when prompted for text generation.

While the aforementioned problem is usually solved by splitting large documents into multiple smaller chunks of text, we unfortunately cannot adopt this method as-is, due to the fact that the medical report documents are composed of several distinct semantic sections, which we must take into consideration when splitting up the text.

Therefore, we investigate how we might extract multiple smaller chunks of text from a single medical report document, such that boundaries between sections are respected and each extracted chunk of text corresponds to exactly one section in the document.

- **RQ2:** How can we use LLMs to perform a domain-specific slot filling task?

The problem at hand represents a structured information extraction challenge, which may typically be solved by for example fine-tuning a pre-trained language model for a named entity recognition (NER) downstream task based on a custom dataset.

Unfortunately, the number of different variables in the information extraction target schema is substantially higher than the total number of medical report documents which have been made available to us by the UKE for the purposes of our work.

Therefore, fine-tuning is practically infeasible due to the high disparity between the number of rows in the training dataset and the number of target labels to predict.

We thus explore instead how we can make use of a pre-trained language model without performing any additional model fine-tuning, specifically by framing the challenge of extracting the desired structured information as a slot filling task.

- **RQ3:** How can we employ prompt engineering techniques to achieve **RQ2** in a zero-shot/few-shot manner?

After making the deliberate decision to frame the information extraction problem at hand as a slot filling task, we further investigate how we can take advantage of in-context learning to overcome the inability to fine-tune a pre-trained model.

Therefore, we explore how we might prompt an LLM to perform our domain-specific slot filling task by providing the LLM with a number of suitable in-context examples.

1.4 System Architecture

We propose the following system for automating the analysis of the medical report documents available at the UKE. The proposed system architecture is comprised of two separate stages. The first stage is responsible for segmenting documents into smaller chunks of text, while respecting the boundaries between different document sections. The second stage is responsible for the actual extraction of the desired structured information from these chunks of text according to the target schema.

Figure 1 illustrates how we solve the problem at hand by building a system that processes medical report documents in the aforementioned two separate stages.

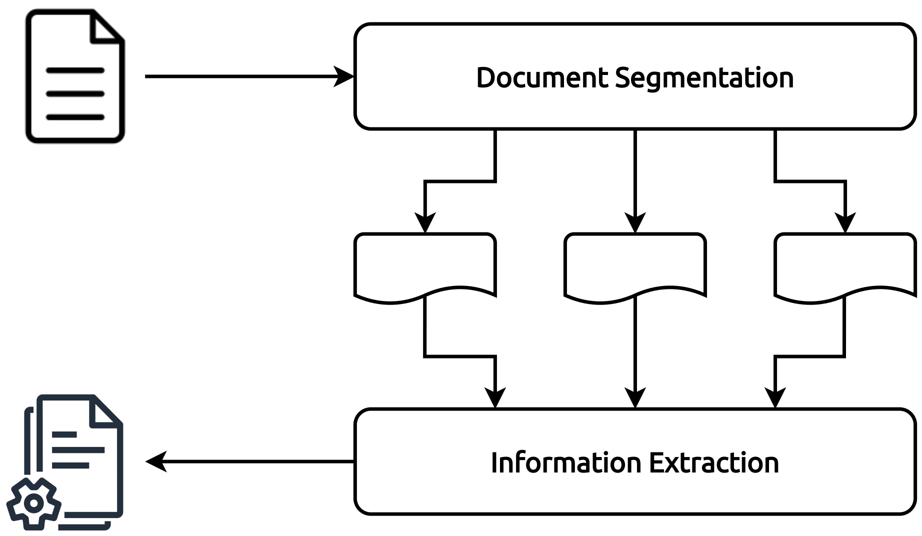


Figure 1: Figure showing the architecture of the proposed document analysis system.

We split the proposed system architecture in two separate stages in order to account for the particular challenges posed by the structure of the medical report documents. Segmenting the documents prior to performing information extraction allows us to first solve **RQ1**, which then further enables us to subsequently solve **RQ2** and **RQ3**.

We describe the concrete implementation details of the first and second stage of the proposed system architecture in Section 3 and Section 4 respectively.

2 Related Work

2.1 Large Language Models

The structured information extraction problem at hand concerns the processing of medical report documents, which frequently contain sensitive patient information.

GPT-4 [2] appears to be the most capable model — which is suitable for solving the problem at hand from a technical standpoint — available at the time of writing.

However, the requirement to process sensitive patient information is incompatible with adopting a closed-source model such as GPT-4 for the purposes of our work.

Accordingly, we investigate what choices are available for LLMs capable of achieving state-of-the-art performance that may be self-hosted on private infrastructure, which is a core requirement in order to comply with relevant data protection regulations.

We find that Vicuna [3], a model originally based on the LLaMA [15] architecture and fine-tuned on conversations from ShareGPT[1], meets the desired requirements.

2.2 Document Understanding

The problem at hand requires the extraction of structured information from medical report documents written in German, which are available as digitally-born PDF documents. After researching common methods for processing such documents, we discover that multi-modal language models constitute the current state of the art.

We therefore consider several different multi-modal model architectures. We briefly discuss whether or not each of them represents a suitable choice, with regards to the particular requirements for document processing imposed by the problem at hand.

LayoutLM [19] is a multi-modal model architecture optimized for processing visually-rich documents. LayoutLM contributes a novel approach that takes advantage of the layout information of visually-rich documents. The LayoutLM architecture uses the BERT [5] architecture as a backbone, while also adding positional layout embeddings during pre-training, as well as adding visual image embeddings during fine-tuning.

[1]ShareGPT.com is a website where users can share real-world conversations with ChatGPT.

LayoutLMv2 [18] subsequently improves upon LayoutLM. LayoutLMv2 differs from LayoutLM in that visual image embeddings are included during pre-training, as opposed to only including the visual image embeddings during the fine-tuning stage.

LayoutLMv3 [7] builds upon the LayoutLMv2 architecture and contributes further improvements. Huang et al. [7] demonstrate that LayoutLMv3 is capable of achieving state-of-the-art performance across various document understanding benchmarks.

One noteworthy commonality between all of the aforementioned model architectures is that these models were exclusively pre-trained on English-language documents. Unfortunately, this makes all of these models unsuitable for solving the problem at hand, considering that processing German-language documents is a core requirement.

Xu et al. [20] introduce LayoutXLM, a multi-modal model architecture for multi-lingual document understanding based on the LayoutLMv2 architecture. LayoutXLM extends the LayoutLMv2 architecture by substituting a cross-lingual textual model and pre-training on a diverse corpus of multi-lingual visually-rich documents [20].

Wang et al. [17] introduce the novel Language-independent Layout Transformer (LiLT) architecture. LiLT decouples visual and textual information during pre-training and re-couples visual and textual information during fine-tuning. LiLT is therefore able to learn the visual layout knowledge from mono-lingual structured documents and then generalize it later to deal with multi-lingual documents [17].

Wang et al. demonstrate that LiLT outperforms LayoutXLM when LiLT is fine-tuned with a mono-lingual textual model, as well as when LiLT is fine-tuned with the same cross-lingual textual model that is used during the pre-training of LayoutXLM [17].

These findings strongly indicate that LiLT is a superior alternative to LayoutXLM, especially when considering our language-specific document processing requirements.

2.3 Slot Filling

The problem at hand requires the extraction of structured information according to the well-known target schema we describe in Section 1.2. We note that the challenge of determining a corresponding value for each variable in the target schema from the text of a given medical report document is naturally reminiscent of a slot filling task.

Accordingly, we research the current state of the art on the subject of slot filling approaches to structured information extraction, and find that the framing as a slot filling task presents a beneficial choice with regards to solving the problem at hand.

Glass et al. [6] introduce an approach for zero-shot slot filling through the use of a novel retrieval architecture, consisting of a fine-tuned dense passage retrieval (DPR) component and a fine-tuned retrieval augmented generation (RAG) component.

Glass et al. build on the prior work of Petroni et al. [11] and show that fine-tuning both components of the retrieval architecture can complement the natural language comprehension capabilities of LLMs, which enable zero-shot inference for slot filling.

Papanikolaou et al. [10] build on the prior work of Glass et al. and adopt zero-shot slot filling to the biomedical domain using a similar fine-tuned retrieval architecture.

Papanikolaou et al. state that building a task-specific dataset for solving structured information extraction problems in the biomedical domain is especially challenging.

Papanikolaou et al. explain that sourcing training data for the biomedical domain is an expensive process, due to the inherent complexity of the domain and the fact that the help of subject matter experts is required for annotating the training dataset.

Papanikolaou et al. thus claim that adopting a traditional information extraction pipeline consisting of separate named entity recognition (NER), entity linking (EL) and relation extraction (RE) steps for the biomedical domain is particularly difficult.

Papanikolaou et al. instead propose applying a slot filling approach to the problem of structured information extraction in the biomedical domain. Papanikolaou et al. argue that the problem of slot filling is similar to open-domain question answering, where reading comprehension is performed over a number of candidate documents.

While Papanikolaou et al. show that fine-tuning the retriever is beneficial for slot filling in the biomedical domain, zero-shot inference is notably possible because the reader model can fill slots not seen during training through reading comprehension.

Papanikolaou et al. therefore conclude that the use of a reader model for natural language comprehension avoids the need for building a task-specific training dataset.

We take some inspiration from Glass et al. and recognize that the particular kind of structured information extraction challenge posed by the problem at hand may be framed as a slot filling task. However, the concrete way in which we perform slot filling is different from the task described by Glass et al. and Papanikolaou et al.

Glass et al. and Papanikolaou et al. both perform slot filling by retrieving multiple candidate documents, any of which may contain the desired answer for the given slot. Therefore, zero-shot here notably refers to the fact that no additional examples for the concrete given slot are provided neither during training nor at inference time.

We instead interpret slot filling as the challenge of extracting the desired answer for a given slot from a specific document, while using additional documents as examples.

While the works of Glass et al. and Papanikolaou et al. thus implement slot filling for the purpose of structured knowledge base population, we take a more classic view of the problem of slot filling, which is instead inspired by frame-based dialog systems.

We further take inspiration from Papanikolaou et al. and recognize that we may leverage the capabilities of LLMs for natural language comprehension, in order to circumvent limitations we face regarding the availability of ground truth annotations.

Cuteri [4] applies a slot filling approach to closed-domain question answering in the cultural heritage domain. Cuteri proposes a system architecture where natural language questions are matched against different templates, which declare certain slots to fill based on the input question. The filled slot answers are then used to complete a formal query, which is executed against a structured knowledge base.

We recognize that framing the problem at hand as a slot filling task further enables us to implement a template matching approach, inspired by the work of Cuteri.

Like Cuteri, we determine the slots to fill for a given system input by means of template matching. However, Cuteri concretely employs slot filling as a method to complete a formal query, which is then executed against a structured knowledge base.

We however use template matching for selecting suitable in-context examples, in order to perform slot filling over medical report documents using an LLM. The filled slot answers thus represent the result of extracting the desired structured information.

2.4 Few-Shot Learning

We realize that fine-tuning a pre-trained language model for solving the problem at hand is not technically feasible, considering the relatively low number of documents available for training and the high number of different variables in the target schema.

Therefore, we research how few-shot learning may allow us to overcome this limitation through the use of in-context examples for improving the quality of model answers.

Brown et al. [1] demonstrate that including examples in the prompt can significantly improve the quality of model answers. Brown et al. thus show that LLMs are capable of few-shot in-context learning. Notably, Brown et al. thereby present an approach to improve the quality of model answers that does not require updating the gradient of a pre-trained language model by fine-tuning on a task-specific training dataset.

However, Brown et al. primarily focus on exploring the difference between zero-shot, one-shot and few-shot settings for a number of different tasks. Brown et al. thus do not put any emphasis on investigating different strategies for in-context example selection. Critically, Brown et al. evaluate the different in-context learning settings for the majority of the experiments by selecting the in-context examples at random.

Liu et al. [8] investigate different strategies for selecting suitable in-context examples, specifically intending to address the aforementioned shortcomings of Brown et al.

Liu et al. subsequently show that using semantic similarity in the embedding space provides a effective strategy for selecting in-context examples. Liu et al. further observe that the effectiveness of such strategies for example selection largely depends on the choice of the particular sentence encoder module used for example retrieval.

Liu et al. conclude by stating the optimal retrieval performance requires fine-tuning the sentence encoder used for computing embeddings on a task-specific dataset.

We therefore understand that embedding-based retrieval constitutes the current state of the art with regards to strategies for selecting in-context examples. Unfortunately, we have reason to believe that embedding-based retrieval as a strategy for selecting suitable in-context examples would be difficult to adopt for the purposes of our work.

Firstly, fine-tuning a sentence encoder is not feasible due to the small size of the corpus of training documents available to us for the purposes of our work. Therefore, we would likely have to forgo optimal retrieval performance, according to Liu et al.

Secondly, even without performing additional fine-tuning on a task-specific dataset, we would still require some suitable sentence encoder to be available off-the-shelf.

However, few models implementing the requisite *sentence transformer* architecture appear to be available off-the-shelf for the language and problem domain relevant for the purposes of our work. Specifically, there are practically no mono-lingual German models compatible with the *sentence transformer* architecture and fine-tuned on the medical domain available off-the-shelf on the HuggingFace model hub as of July 2023.

While some cross-lingual models implementing the *sentence transformer* architecture are available off-the-shelf, we believe that there is a general lack of available mono-lingual models fine-tuned on particular domains, other than for the English language.

We therefore argue based on this perceived lack of choice, that there is sufficient reason to investigate alternative strategies for selecting suitable in-context examples, which do not employ semantic similarity in the embedding space as a retrieval metric.

3 Document Segmentation

The following chapter describes the implementation of the first stage of the proposed system architecture. The first stage of the system concerns the processing of medical report documents such that the text of the documents is split into multiple smaller chunks of text. The next chapter explains the second stage of the system, where the chunks of text yielded by the first stage are further processed for document analysis.

Before we can split up the text of a document into smaller chunks, we first need to segment a medical report document into the distinct semantic sections it is comprised of. The segmentation of a document is a necessary prerequisite to chunking, as we must respect the section boundaries within a document when splitting up the text.

The following chapter therefore primarily focuses on the implementation details of the document segmentation approach. The chapter does not put emphasis on discussing how the resulting text is then split up into smaller chunks, as chunking plain text is a comparatively trivial problem for which many off-the-shelf solutions are available.

3.1 Approach

We frame the problem of segmenting a medical report document into the distinct semantic sections it is comprised of as a named entity recognition task and opt to fine-tune a multi-modal model to classify the individual words in the text of a document based on the role of the word with regards to the visual structure of the document.

We then use the predictions of the model to determine which of the words in the text of a given document are part of the heading or body text of one of the distinct semantic sections within a document. We can thus simply ignore any other words in the text — such as words that are part of page headers and footers — which allows us extract only the text from the relevant semantic sections within the document.

3.2 Model Architecture

We fine-tune a model of the Language-independent Layout Transformer (LiLT) [17] architecture, because the problem at hand requires processing documents written in German, as well as because LiLT is capable of achieving state-of-the-art performance.

Notably, the LiLT architecture is centered around the idea of combining a pre-trained visual model with any textual model implementing the RoBERTa [9] architecture during fine-tuning. LiLT may thus be adopted for processing visually-rich documents written in any language, as long as a compatible textual model is available [17].

Scheible et al. [12] introduce GottBERT, the first mono-lingual German model based on the RoBERTa architecture. Scheible et al. further show that GottBERT is capable of outperforming a cross-lingual RoBERTa model on several German-language tasks.

We therefore decide to use GottBERT — specifically the *gottbert-base*[2] release of GottBERT available on HuggingFace — as the textual model during fine-tuning.

3.3 Training Dataset

In this section, we describe the steps taken to prepare a dataset for model fine-tuning based on the training corpus of medical report documents provided to us by the UKE.

3.3.1 Data Collection

The corpus of training documents to be used for model fine-tuning consists of 151 individual medical reports describing hospitalization cases of patients at the UKE.

The documents in the training corpus are provided to us by the UKE in a state where sensitive patient information has been manually redacted through a process where filled black rectangles are drawn over any area on a page in a medical report document where sensitive information is mentioned in the text of the given document.

3.3.2 Pre-Processing

Building a suitable dataset for fine-tuning a LiLT model from the corpus of training documents requires several pre-processing steps due to the format of the documents.

The documents that make up the training corpus are originally provided to us in the form of a number of digitally-born PDF files. Notably, each of the original files contains one or more individual medical report documents — all of which are related to the same distinct patient case — that have been combined into a single PDF file.

[2]`https://huggingface.co/uklfr/gottbert-base`

Because many of these original PDF files contain more than one medical report document, we first have to split the original files and create one separate PDF file for each of the individual medical report documents.

We solve this problem by programmatically reading the original PDF files page by page and employing a heuristic to determine whether a given page is the first page in a medical report document, which is trivial since certain words are known to only appear on the first page, but not on any other page in a medical report document.

We continue by processing these separate files we create for each of the individual medical report documents and converting every page in the document to an image.

We then use Tesseract [13], an open-source optical character recognition (OCR) engine, to extract the textual contents found on the page for each of these images. The output of Tesseract notably includes both the actual textual contents and the positional information about where each word in the text of appears on the image.

However, before we can extract the text found on the page from one of these images using Tesseract, we slightly need to pre-process the page images due to a problem that is caused by the method which was used to redact sensitive patient information. Tesseract appears to be consistently unable to recognize some of the text located in the close vicinity of a black rectangle that has been drawn over an area on the page.

We solve this problem by applying basic morphological image processing techniques to detect where in the image of a page the filled black rectangles are located and manually change the fill color of the rectangles to match the white page background.

3.3.3 Human Annotation

Because we decide to fine-tune a model of the LiLT architecture, we need to create the annotations for the training dataset in a format where we do not just annotate the words in the text, but rather pairs consisting of both the individual words and the corresponding positional information about where each word appears on the page.

Unfortunately, the popular tools that are commonly used for dataset annotation do not support creating the type of annotations that is required by the LiLT architecture.

We solve this problem by building a custom pipeline for creating annotations in the required format. We first use existing tools for image annotation in order to obtain annotations consisting of labelled regions for all of the individual page images.

We then align these annotations with the textual and positional information we have extracted from the page images using Tesseract. We derive the labels for the individual words by using the common positional information to determine which of the labelled image regions the individual words intersect, for every single page image.

We decide to use Label Studio [14], an open-source software for dataset annotation, for the image annotation process because it is free to use and can be run locally — which is desirable considering the constraints of our work — and provides an easy to use editor for creating spacial image annotations in the aforementioned manner.

We use the following classes for annotating regions of text within the page images.

- **section:** The name of a section heading appearing at the start of each section.
- **header:** The entire text in the letterhead on the first page and the header containing the page number on every subsequent page.
- **footer:** The entire text in the footer of the first page.
- **opening:** The entire text of the "opening" paragraph on the first page.
- **closing:** The entire text of the "closing" paragraphs on the final pages.

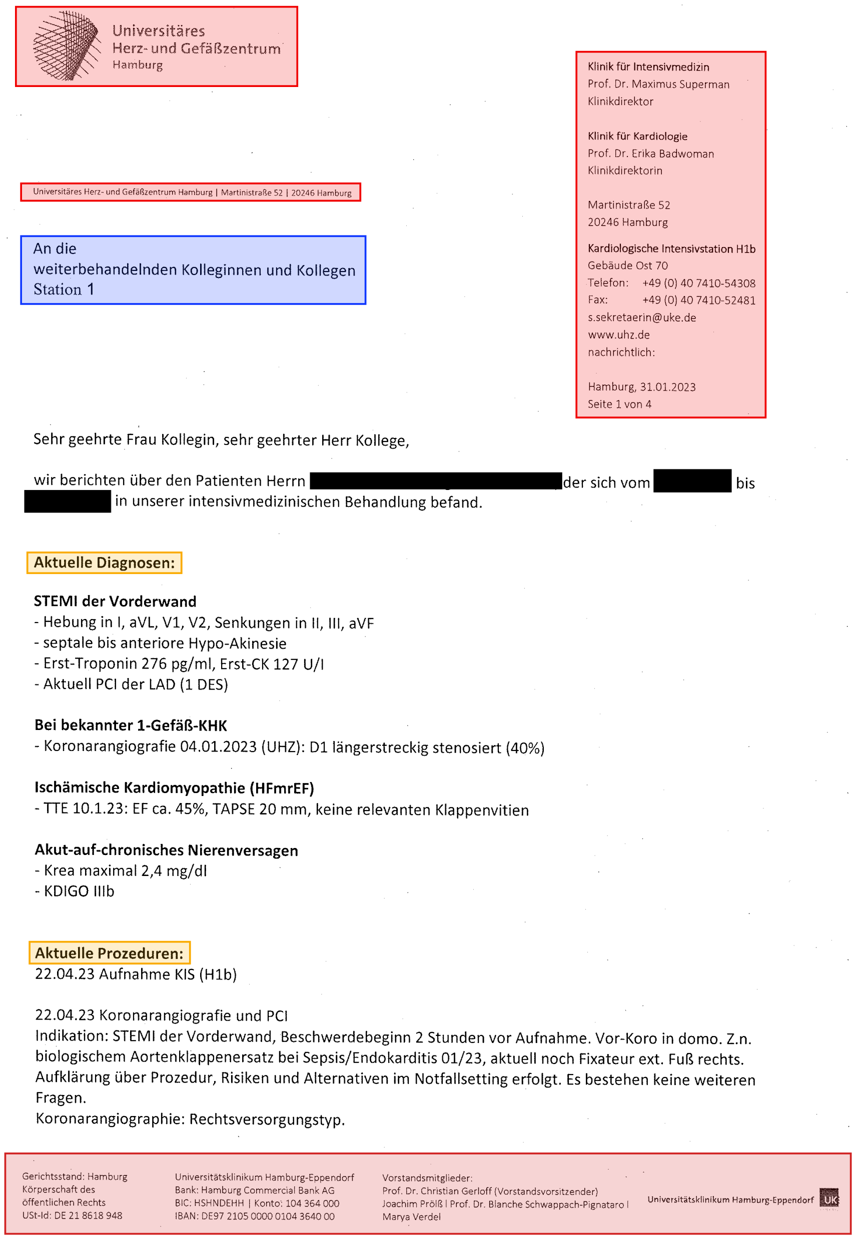
Figure 2 shows an illustration of the annotations created for the first page in a medical report document with the labelled regions matching the structure of the document.

3.3.4 Considerations

We must take special care when generating the three different *train*, *test* and *dev* dataset splits due to the fact that words annotated with some class labels do not appear with an equal frequency across the different types of pages in a document.

While words annotated with the **section** class label may — in principle — appear on any page in a document, the same is not the case for the remaining class labels.

The vast majority of all words annotated with the **header** class label, for example, appears on the first page of a document — which always contains the boilerplate letterhead — while a comparatively small percentage of the words annotated with the **header** class label accounts for the page number present on all subsequent pages.



Universitäres
Herz- und Gefäßzentrum
Hamburg

Universitäres Herz- und Gefäßzentrum Hamburg | Martinistraße 52 | 20246 Hamburg

An die
weiterbehandelnden Kolleginnen und Kollegen
Station 1

Klinik für Intensivmedizin
Prof. Dr. Maximus Superman
Klinikdirektor

Klinik für Kardiologie
Prof. Dr. Erika Badwoman
Klinikdirektorin

Martinistraße 52
20246 Hamburg

Kardiologische Intensivstation H1b
Gebäude Ost 70
Telefon: +49 (0) 40 7410-54308
Fax: +49 (0) 40 7410-52481
s.sekretaerin@uke.de
www.uhz.de
nachrichtlich:

Hamburg, 31.01.2023
Seite 1 von 4

Sehr geehrte Frau Kollegin, sehr geehrter Herr Kollege,

wir berichten über den Patienten Herrn [redacted] der sich vom [redacted] bis [redacted] in unserer intensivmedizinischen Behandlung befand.

Aktuelle Diagnosen:

STEMI der Vorderwand
- Hebung in I, aVL, V1, V2, Senkungen in II, III, aVF
- septale bis anteriore Hypo-Akinesie
- Erst-Troponin 276 pg/ml, Erst-CK 127 U/l
- Aktuell PCI der LAD (1 DES)

Bei bekannter 1-Gefäß-KHK
- Koronarangiografie 04.01.2023 (UHZ): D1 längerstreckig stenosiert (40%)

Ischämische Kardiomyopathie (HFmrEF)
- TTE 10.1.23: EF ca. 45%, TAPSE 20 mm, keine relevanten Klappenvitien

Akut-auf-chronisches Nierenversagen
- Krea maximal 2,4 mg/dl
- KDIGO IIIb

Aktuelle Prozeduren:
22.04.23 Aufnahme KIS (H1b)

22.04.23 Koronarangiografie und PCI
Indikation: STEMI der Vorderwand, Beschwerdebeginn 2 Stunden vor Aufnahme. Vor-Koro in domo. Z.n. biologischem Aortenklappenersatz bei Sepsis/Endokarditis 01/23, aktuell noch Fixateur ext. Fuß rechts. Aufklärung über Prozedur, Risiken und Alternativen im Notfallsetting erfolgt. Es bestehen keine weiteren Fragen.
Koronarangiographie: Rechtsversorgungstyp.

Gerichtsstand: Hamburg
Körperschaft des
öffentlichen Rechts
USt-Id: DE 21 8618 948

Universitätsklinikum Hamburg-Eppendorf
Bank: Hamburg Commercial Bank AG
BIC: HSHNDEHH | Konto: 104 364 000
IBAN: DE97 2105 0000 0104 3640 00

Vorstandsmitglieder:
Prof. Dr. Christian Gerloff (Vorstandsvorsitzender)
Joachim Prölß | Prof. Dr. Blanche Schwappach-Pignataro |
Marya Verdel

Universitätsklinikum Hamburg-Eppendorf UKE

Figure 2: Figure showing an example of the annotations created for an image of the first page from a fictional medical report document using image annotation software.

Further, all of the words annotated with the **opening** and the **footer** class labels exclusively appear on the first page in a document, while all the words annotated with the **closing** class label exclusively appear on the final few pages in a document.

The first page in a document in particular naturally only accounts for a single page per medical report document in the training corpus. Accordingly, the total number such pages also only makes up a small percentage of all pages in the training dataset.

Because each row in the training dataset represents a single page extracted from a medical report document in the training corpus, we must thus be careful to consider the type of page when distributing the pages between the *train*, *test* and *dev* splits.

Thankfully, we can trivially implement an intervention for making sure that we include an equal proportion of every type of page in each of the three dataset splits.

Because we know that the **opening** and **footer** class labels always appear on the first page in a document, and that the **closing** class label always appears on the final few pages in a document, we can easily detect and consider the type of a given page when generating the three different *train*, *test* and *dev* splits of the training dataset.

3.4 Experiments

The following section describes the steps taken to fine-tine a LiLT model on the dataset described earlier and investigates the performance of the fine-tuned model.

3.4.1 Training Setting

We fine-tune the model for 2500 steps with a learning rate of $1e - 5$ on a single NVIDIA GeForce RTX 3080 Laptop GPU with 16 GiB of total GPU memory.

Note that we use the overall F1 score as a metric for selecting the best model during training due to the general imbalance between the different labels in the dataset.

3.4.2 Evaluation

After fine-tuning has completed, we evaluate the resulting model checkpoint against the *test* split of the dataset, which exclusively contains documents not seen during training. Table 1 shows the evaluation results of the fine-tuned model checkpoint.

Label	**Precision**	**Recall**	**F1 score**
section	0.9407	0.9737	0.9569
header	0.9902	0.9806	0.9854
footer	0.9524	0.9524	0.9524
opening	0.7576	0.8621	0.8065
closing	1.0000	1.0000	1.0000
overall	0.9410	0.9672	0.9539

Table 1: Table showing the evaluation statistics of the fine-tuned LiLT model.

We can clearly observe based on the results shown in Table 1 that the majority of the different class labels perform quite well, with all but one of the labels exceeding an F1 score of 95% during the evaluation of the fine-tuned model checkpoint.

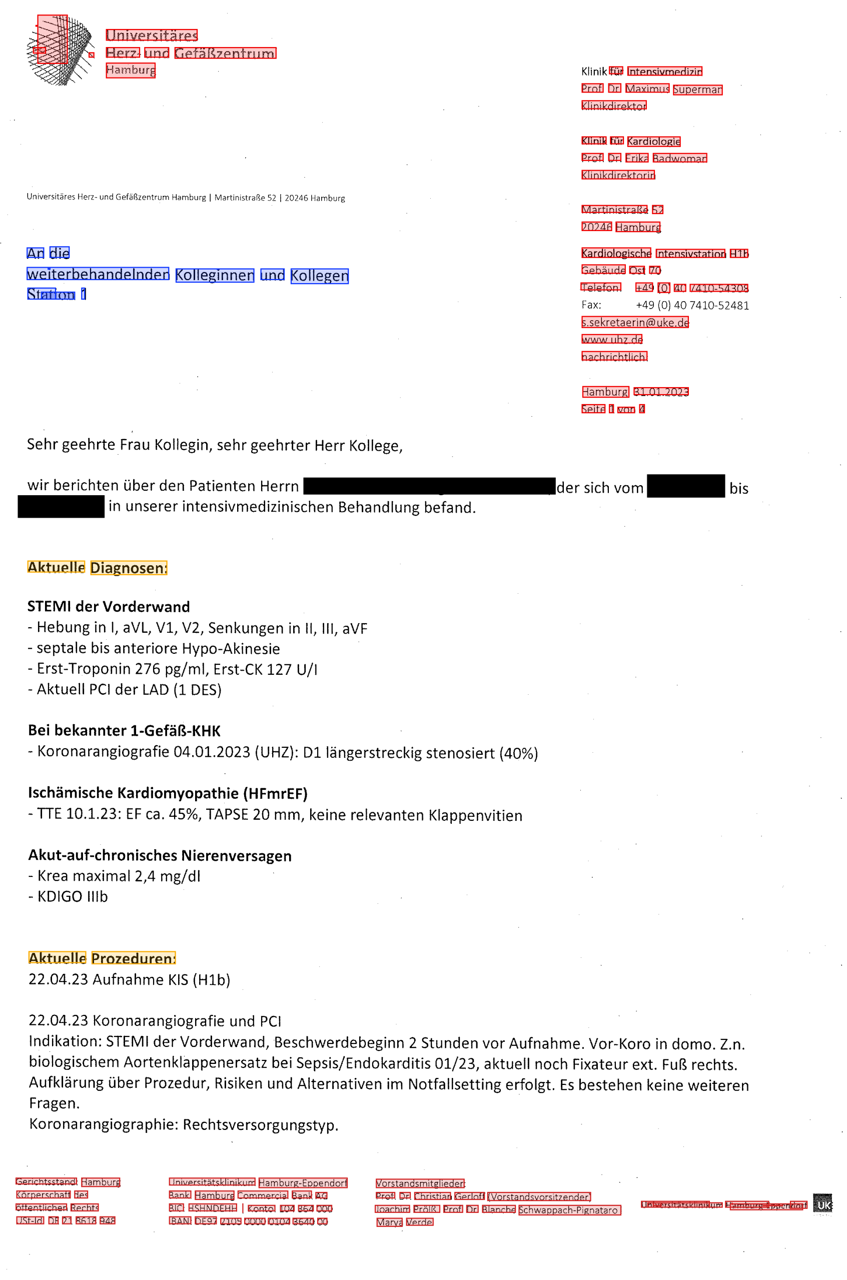
The major outlier in this regard appears to be the **opening** class label, which shows an evaluation F1 score of only around 80% and a precision score of only around 75%.

We attempt to explain this finding by posing the hypothesis that the comparatively poor performance of the **opening** class is likely caused by the fact that the text in the medical report documents annotated with this label tends to be relatively heterogeneously phrased, unlike the text annotated with the remaining class labels, which generally appears to be phrased much more homogeneously in comparison.

Consequently, we test this hypothesis by noting that, if true, the evaluation score of a class label should increase across all metrics as the variation in the phrasing of the correspondingly annotated text of the documents in the training dataset decreases.

We find that this in fact appears to be the case, as the **closing** class label, for which the annotated text is virtually identical across the many different documents in the training dataset — due to the standard boilerplate language that is used for this part of the text — coincidentally also reaches the highest evaluation performance.

Similarly, we can observe that the **header** class label, for which the annotated text also largely consists of standard boilerplate language shared between the different documents in the dataset, as well as the **section** class label, for which the annotated text is exclusively comprised a small set of well-known document section names, coincidentally reach the second and third highest evaluation performance respectively across all metrics, which is further evidence in support of the stated hypothesis.



Universitäres
Herz- und Gefäßzentrum
Hamburg

Klinik für Intensivmedizin
Prof. Dr. Maximus Superman
Klinikdirektor

Klinik für Kardiologie
Prof. Dr. Erika Badwoman
Klinikdirektorin

Martinistraße 52
20246 Hamburg

Kardiologische Intensivstation H1b
Gebäude Ost 70
Telefon: +49 (0) 40 7410-54308
Fax: +49 (0) 40 7410-52481
s.sekretaerin@uke.de
www.uhz.de
nachrichtlich:

Hamburg, 31.01.2023
Seite 1 von 4

Universitäres Herz- und Gefäßzentrum Hamburg | Martinistraße 52 | 20246 Hamburg

An die
weiterbehandelnden Kolleginnen und Kollegen
Station 1

Sehr geehrte Frau Kollegin, sehr geehrter Herr Kollege,

wir berichten über den Patienten Herrn [redacted] der sich vom [redacted] bis [redacted] in unserer intensivmedizinischen Behandlung befand.

Aktuelle Diagnosen:

STEMI der Vorderwand
- Hebung in I, aVL, V1, V2, Senkungen in II, III, aVF
- septale bis anteriore Hypo-Akinesie
- Erst-Troponin 276 pg/ml, Erst-CK 127 U/l
- Aktuell PCI der LAD (1 DES)

Bei bekannter 1-Gefäß-KHK
- Koronarangiografie 04.01.2023 (UHZ): D1 längerstreckig stenosiert (40%)

Ischämische Kardiomyopathie (HFmrEF)
- TTE 10.1.23: EF ca. 45%, TAPSE 20 mm, keine relevanten Klappenvitien

Akut-auf-chronisches Nierenversagen
- Krea maximal 2,4 mg/dl
- KDIGO IIIb

Aktuelle Prozeduren:
22.04.23 Aufnahme KIS (H1b)

22.04.23 Koronarangiografie und PCI
Indikation: STEMI der Vorderwand, Beschwerdebeginn 2 Stunden vor Aufnahme. Vor-Koro in domo. Z.n. biologischem Aortenklappenersatz bei Sepsis/Endokarditis 01/23, aktuell noch Fixateur ext. Fuß rechts. Aufklärung über Prozedur, Risiken und Alternativen im Notfallsetting erfolgt. Es bestehen keine weiteren Fragen.
Koronarangiographie: Rechtsversorgungstyp.

Gerichtsstand: Hamburg
Körperschaft des
öffentlichen Rechts
USt-IdNr. DE 21 8618 948

Universitätsklinikum Hamburg-Eppendorf
Bank: Hamburg Commercial Bank AG
BIC: HSHNDEHH | Konto: 104 364 000
IBAN: DE97 2105 0000 0104 3640 00

Vorstandsmitglieder:
Prof. Dr. Christian Gerloff (Vorstandsvorsitzender)
Joachim Prölß, Prof. Dr. Blanche Schwappach-Pignataro
Marya Verde

Universitätsklinikum Hamburg-Eppendorf

Figure 3: Figure illustrating the result of running inference over the first page from a fictional example medical report document using the fine-tuned model checkpoint.

Figure 3 shows the result of using the fine-tuned model checkpoint to run inference over the same first page from a fictional medical report document shown in Figure 2.

3.5 Discussion

RQ1 concerns the question of how we can overcome the limitations of LLMs with regards to the maximum number of input tokens supported at a time during inference.

We circumvent this particular issue by fine-tuning a multi-modal language model to classify the different parts of the text within a medical report document according the role of the text with regards to the common visual structure of the documents.

Notably, we use the predictions of the fine-tuned model to create chunks only from the text of the relevant semantic sections within a given medical report document.

The evaluation results for the fine-tuned model checkpoint shown in Table 1 and the inference example illustrated in Figure 3 clearly demonstrate that the solution we implement for addressing the first research question works successfully in practice.

4 Information Extraction

The following chapter discusses the second stage of the proposed system architecture and the implementation thereof, as well as relevant considerations and limitations.

The first stage of the proposed systems yields chunks created from the text in the individual semantic sections within a given medical report document. The second stage concerns the extraction of the desired structured information from these chunks.

4.1 Approach

We frame the problem of actually extracting the desired structured information from a given medical report document as a slot filling task. According to this framing, each of the target variables, for which we want to extract information from the documents, represents a slot that may be filled with a corresponding answer. The corresponding answer for each of these slots must be determined based on the text from one or more of the distinct semantic sections present within a medical report document.

We perform the actual filling of the slots by incrementally prompting an LLM with small chunks of text extracted from the different sections within a given document.

The aforementioned process allows us to extract structured information from a whole document by performing slot filling over all of its relevant constituent parts and then aggregating the individual answers obtained for the slots in the well-known schema.

4.2 Model Architecture

We use an instance of Vicuna 1.5[3] hosted on infrastructure owned and operated by the University of Hamburg throughout the experiments we describe in this chapter.

The particular instance of the model that is available to us for the purposes of our work is an 8-bit quantized variant of the *vicuna-13b-v1.5-16k*[4] release of Vicuna 1.5.

[3]Vicuna 1.5 is based on the Llama 2 [16] architecture, while prior versions of the model are based on the LLaMA [15] architecture. See `https://github.com/lm-sys/FastChat/blob/main/docs/vicuna_weights_version.md` for more information regarding the different versions of Vicuna.

[4]`https://huggingface.co/lmsys/vicuna-13b-v1.5-16k`

4.3 Prompt Example Pre-Retrieval

We decide to include a number of suitable in-context examples when prompting the LLM to perform our domain-specific slot filling task. We are thereby able to take advantage of in-context learning in lieu of fine-tuning a pre-trained language model.

Therefore, whenever we prompt the LLM to perform slot filling for a given chunk of text extracted from a section within a medical report document — which we will call the *query chunk* — we also include a number of other chunks of text extracted from documents in the training corpus — which we will call the *example chunks* — to serve as in-context examples that are suitable for the particular given query chunk.

Naturally, choosing in-context learning as a paradigm requires us to employ some strategy for determining which of the example chunks available within the corpus of training documents are suitable as in-context examples for a particular query chunk.

The current state-of-the-art solution for selecting suitable in-context examples is to use semantic similarity in the embedding space as distance metric for determining which of the available examples are semantically close to the given query context.

While this strategy for example selection has been shown to work well in the space of open-domain question answering tasks, the retrieval of in-context examples based on embeddings comes with practical downsides with regards to the problem at hand.

We refer to Section 2.4 for a longer discussion on the advantages of embedding-based retrieval of in-context examples and the current state of the art, as well as an explanation on why this approach may be difficult to adopt for the problem at hand.

Fortunately, the deliberate framing as slot filling task makes the problem at hand different from generic open-domain question answering tasks — where embedding-based retrieval is commonly used for example selection — in one important aspect.

Whenever we prompt the LLM to perform slot filling over a given query chunk, we are capable of knowing which of the distinct semantic sections within a medical report document the chunk was extracted from, as we can easily preserve this information.

Further, for every slot derived from a variable in the well-known target schema, we know *a priori* which sections in a document the slot in question may be filled from.

When taking these facts into consideration together, one major implication becomes apparent, presenting an alternative to embedding-based in-context example retrieval.

Looking at the problem of generic open-domain question answering, the query context — that is, the question to be answered — is naturally only known at inference time. Therefore, any in-context examples to include when prompting the LLM must necessarily be chosen at inference time as well, and cannot possibly be chosen earlier.

However, when framing the problem at hand as a slot filling task, the same limitation with regards to when the selection of in-context examples may occur does not apply.

On the one hand, the concrete text of any particular query chunk is still only known at inference time. On the other hand, however, we can know ahead of inference time which subset of slots may potentially be filled from any given chunk, if we preserve the information about which section in the document the chunk was extracted from.

We can therefore employ a strategy that chooses examples exclusively based on the suitability of an example to one of the distinct semantic sections present in every medical report document, because we assume that these particular examples will also be suitable to any concrete chunks of text extracted from the same semantic section.

The corpus of training documents made available to us for the purposes of our work includes "gold" slot answers for every medical report document. However, we do not actually know which section within a given document any of the provided "gold" slot answers are extracted from. At the same time, the slots derived from the variables in the target schema are generally fillable from more than one document section.

We thus first need to solve the problem of deriving the correct "gold" slot answer for the individual chunks of text extracted from the different sections within the documents in the training corpus. Otherwise, we cannot employ any in-context learning technique at all, as we require the "gold" slot answers to be known for each example chunk, not just the whole document the example chunk was extracted from.

We solve this problem by using hand-crafted regular expressions to label all chunks of text extracted from the documents in the training corpus. The labels we assign to a particular chunk correspond to the slots we believe to be fillable from the chunk in question. We then use the assigned labels to determine which of the "gold" slot answers available for the whole document are relevant for the chunk in question.

Notably, the problem of having to derive the "gold" slot answers for the individual example chunks applies regardless of how specifically we select the concrete in-context examples, including in the case that we were to employ embedding-based retrieval.

We further want to point out that since we know *a priori* which slots may be fillable for a given query chunk, any example chunks selected for a particular query chunk using some embedding-based retrieval strategy would necessarily be a subset of the example chunks that we are able to select purely based on the available slot labels.

Therefore, we propose to directly use the available slot labels for determining which example chunks are suitable as in-context examples for a given group of slots to fill.

We admit that example chunks selected at inference time using some embedding-based retrieval strategy may be more suitable for the concrete given query chunks, compared to selecting example chunks ahead of inference for a group of slots to fill.

More concretely, using embedding-based retrieval at inference time might yield in-context examples that are more semantically similar to the query chunk. However, due to the aforementioned problems with adopting embedding-based retrieval for the purposes of our work, we believe that it is reasonable to forgo hypothetical retrieval performance benefits and instead implement a more practical selection strategy.

We thus decide to implement a strategy for selecting example chunks suitable for a given group of slots to fill ahead of inference time based on the available slot labels.

4.4 Training Dataset

In this section, we describe the steps taken to build a training dataset for the custom prompt example pre-retrieval method we implement based on the proposed approach.

4.4.1 Pre-Processing

We start by using the fine-tuned LiLT model for document segmentation to run inference over the entire training corpus of medical report documents, yielding the extracted section name and the whole text body for each section in every document.

We continue by iterating over each of the section name and body text pairs from the last step and splitting the whole section body into multiple smaller chunks of text.

We then use the heuristic tagging procedure described below to assign labels to the individual chunks of text, based on which of the slots in the target schema we believe to be fillable from the text of the given chunk, according to the tagging results.

We rely on the "gold" slot data to automatically detect and remove false positive labels assigned based on the tagging results by dropping any label for a chunk of text that was extracted from document where the respective slot does not have a positive ground truth answer, since we know based on the "gold" slot data that the slot in question may not actually be filled from any section in this particular document.

We are additionally able to remove some of the remaining false positive labels due to the fact that we know which of the distinct semantic sections within a document each of the slots in the target schema may potentially be filled from, allowing us to detect whether the assignment of a label to a given chunk of text is clearly wrong, based on checking whether the section from which the given chunk was extracted from is among the sections for which the slot in question is known to be fillable from.

After removing as many false positive labels as possible, we finally generate the three different *train*, *dev* and *test* splits of the dataset. Notably, we take care to create the different dataset splits such that each row in the *train* split consists only of a single chunk of text and the relevant labels, while the rows in the *test* and *dev* splits represent whole documents containing the original text for each of the extracted sections as well as the complete "gold" slot data available for the entire document.

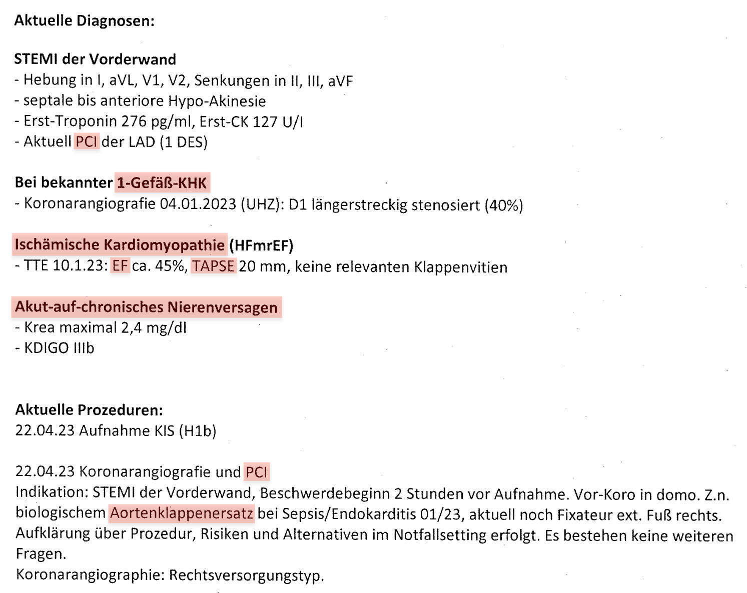
This difference in the shape of the dataset rows between the three different splits is required as a consequence of how the dataset is used during training, where individual chunks of text are chosen as in-context examples based on the assigned labels and the semantic document section the chunk was extracted from, while the resulting choice of examples is evaluated against the entire text of the relevant document sections.

4.4.2 Regular Expression Tagging

We define regular expressions for each slot in the well-known schema by manually reviewing the medical report documents in the training corpus and hand-crafting a regular expression for a particular slot based on the different ways we observe the filler answers for that slot to be worded across all documents in the training corpus.

We refer to the source code repository[5] for a list of the concrete regular expressions.

[5] `https://github.com/semantic-systems/medical-slot-filling`



Aktuelle Diagnosen:

STEMI der Vorderwand
- Hebung in I, aVL, V1, V2, Senkungen in II, III, aVF
- septale bis anteriore Hypo-Akinesie
- Erst-Troponin 276 pg/ml, Erst-CK 127 U/l
- Aktuell PCI der LAD (1 DES)

Bei bekannter 1-Gefäß-KHK
- Koronarangiografie 04.01.2023 (UHZ): D1 längerstreckig stenosiert (40%)

Ischämische Kardiomyopathie (HFmrEF)
- TTE 10.1.23: EF ca. 45%, TAPSE 20 mm, keine relevanten Klappenvitien

Akut-auf-chronisches Nierenversagen
- Krea maximal 2,4 mg/dl
- KDIGO IIIb

Aktuelle Prozeduren:
22.04.23 Aufnahme KIS (H1b)

22.04.23 Koronarangiografie und PCI
Indikation: STEMI der Vorderwand, Beschwerdebeginn 2 Stunden vor Aufnahme. Vor-Koro in domo. Z.n. biologischem Aortenklappenersatz bei Sepsis/Endokarditis 01/23, aktuell noch Fixateur ext. Fuß rechts. Aufklärung über Prozedur, Risiken und Alternativen im Notfallsetting erfolgt. Es bestehen keine weiteren Fragen.
Koronarangiographie: Rechtsversorgungstyp.

Figure 4: Figure showing an illustration of how regular expressions are used to detect instances of relevant keywords in the text of a medical report document.

Figure 4 shows an illustration of how instances of relevant terms and phrases are detected in the text of a medical report document using the hand-crafted regular expressions. For example, we can see that the phrase "3-Gefäß-KHK" is detected in the "Aktuelle Diagnosen" section, which implies that the slot "Coronary artery disease" may be filled with a positive value from the text of the document in question.

While we hereby demonstrate that some part of the structured information extraction problem at hand can be solved simply by relying on regular expressions, we want to explicitly point out why it still is necessary to employ the help of an LLM at all.

Notably, the concrete hand-crafted regular expressions are designed on the basis of the particular text from the specific documents present within the training corpus, and will inevitably result in some false positive and false negative match instances.

Further, only those slots in the well-known target schema of the boolean value data type could even theoretically be filled using regular expressions alone, because the enumeration and text value data types necessitate natural language comprehension.

Therefore, while regular expressions are arguably useful for reducing the overall effort required for annotating the training dataset, we nonetheless need the help of an LLM to successfully solve the structured information extraction problem at hand.

While the described heuristic tagging method is not perfectly accurate, the remaining false positive matches not eliminated by the procedure described earlier, as well as the false negative match instances missed entirely by the concrete regular expressions used during dataset creation are of little consequence for the overall training process.

Notably, a chunk of text assigned a false positive label will inevitably be discarded during training, as the LLM will naturally determine such chunks to be unsuitable to be used as an in-context example for learning how to fill that particular slot, just as chunks with a true positive corresponding label are discarded whenever the LLM determines that another chunk assigned the same label serves as a better in-context example as a consequence of the concrete language used in the text of these chunks.

Further, assuming that a sufficient minimum number of chunks to be used as in-context examples is present in the dataset for every slot in the well-known target schema, false negative match instances are not particularly problematic either.

While the LLM might still learn to fill a particular slot when a chunk not assigned some label is coincidentally included in the prompt due to being selected based on a different label, in this case the chunk in question will necessarily be discarded as the LLM will always mistakenly judge such chunks to be unsuitable in-context examples due to the fact that the expected answer for each in-context example is determined based on the assigned labels only, not the "gold" slot data for the whole document.

4.4.3 Considerations

We must take special care when generating the three different *train*, *test* and *dev* dataset splits due to the imbalanced nature of the "gold" slot data, since the number of the positive ground truth answers found across the medical report documents in the corpus varies wildly between the different slots in the well-known target schema.

We circumvent this issue by generating the different dataset splits in a way where we iterate through all the slots in the target schema in the increasing order of the total number of labels present for that slot across all chunks and subsequently distribute the next three available documents for which this label has been assigned to any chunk extracted from these document evenly across the three different dataset splits.

While this intervention is far from perfect, since picking a document based on a particular label represents a greedy choice which impacts the further distribution of documents based on the remaining labels, we can see that the described strategy works reasonably well in practice as it generally appears to distribute documents across the different dataset splits such that the total number of positive answers for each slot is distributed relatively evenly across the different splits of the final dataset.

We refer to the source code repository[6] for a breakdown of the number of chunks with a particular label for any of the slots in the well-known target schema, which are included in the three different *train*, *test* and *dev* splits of the training dataset.

4.5 Prompt Example Tuning Algorithm

We propose the following algorithm for implementing the prompt example pre-retrieval strategy we introduce and describe, as well as justify in Section 4.3.

The primary goal of the algorithm is to determine the best possible choice of in-context examples for a given group of slots — which we collectively call a *prompt* — from the chunks of text that are available from the *train* split of the training dataset.

Because we select examples ahead of inference time, we have no way to measure the quality of a concrete chunk of text as an in-context example for a given slot *a priori*.

We solve this problem by implementing a semi-random greedy search over the space of all possible combinations of the available examples for every slot in the prompt.

More specifically, we search for a locally optimal choice of examples by evaluating a subset of all possible combinations of example chunks available from the *train* split of the dataset against the medical report documents from the *dev* split of the dataset.

The algorithm consists of two separate phases. Figure 5 shows a high-level illustration of the abstract algorithm logic and the relationship between the two separate phases.

The first phase of the algorithm, rotation, concerns finding the best possible example for every slot in the prompt from the available example chunks.

The second phase of the algorithm, minification, concerns further improving the choice of examples yielded by the rotation phase by strategically removing examples.

[6]`https://github.com/semantic-systems/medical-slot-filling`

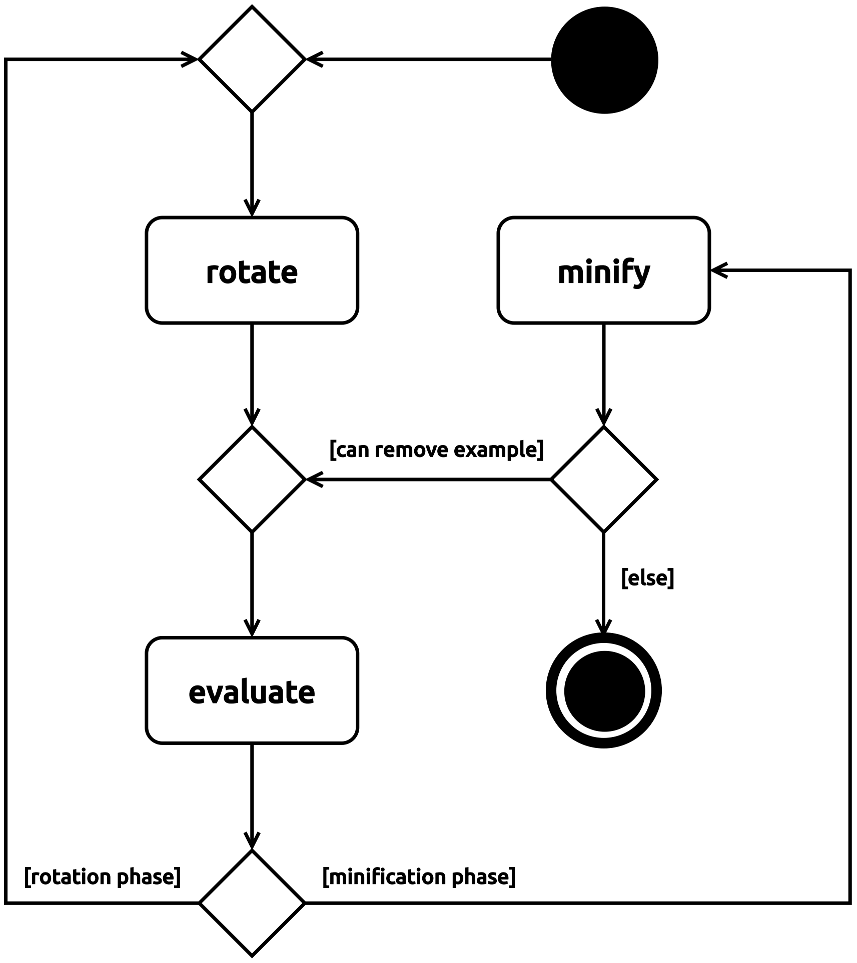


Figure 5: Figure showing an illustration of the abstract logic of the algorithm we propose, as well as the relationship between the two separate phases of the algorithm.

During each iteration of the algorithm, the current choice of examples is modified and subsequently evaluated against documents from the *dev* split of the training dataset. The algorithm finally terminates once it has found the locally optimal choice of examples for every slot in the prompt according to the overall evaluation results.

We refer to the source code repository[7] for the implementation of the algorithm and for more details regarding the reference implementation of the algorithm logic.

4.5.1 Rotation

The goal of the rotation phase is to find the best choice of examples while at the same time avoiding a brute-force search over all possible combinations of examples.

During each rotation step, the current example chosen for every slot in the prompt is either swapped with new example from the *train* split of the dataset or not modified.

Therefore, the algorithm only ever considers a small subset of the space of all possible combinations of examples by rotating multiple examples at once whenever possible.

The algorithm transitions into the minification phase once none of the examples chosen for any slot in the prompt are modified during the current rotation step.

The current example for every slot in the prompt is swapped with a new example chosen at random from the *train* split of the training dataset, unless the evaluation results from the previous iteration show a perfect score for the current example chosen for the given slot, or no more unseen examples are available for the given slot.

When no more unseen examples are available for a given slot in the prompt, the current example is set to the best example seen so far for the slot in question based on the previous evaluation results and the example for that slot is not further modified.

Notably, we discover during manual experimentation that example chunks with a smaller number of total assigned slot labels appear to generally make a better choice when compared to example chunks with a higher number of total assigned slot labels.

Therefore, we decide to implement the random selection such that examples for a given slot in the prompt are selected at random across all example chunks with a label matching the given slot, but in the increasing order of total assigned slot labels.

[7] `https://github.com/semantic-systems/medical-slot-filling`

Notably, we decide to only consider a subset of all documents from the *dev* split of the training dataset for evaluating of the current choice of examples, in an attempt to reduce the number of LLM inference requests as a cost and time saving measure.

We attempt to pick a representative subset of documents from the *dev* split of the training dataset by always including all available documents with positive "gold" slot answers for any of the slots in the prompt, as well as an equal number of the remaining documents without positive "gold" slot answers for any slot in the prompt.

The evaluation of the choice of examples is subsequently performed by iteratively prompting the LLM with the current choice of example chunks, against the relevant query chunks extracted from the documents in the *dev* split of the training dataset.

The algorithm importantly limits the evaluation to include only the query chunks extracted from any semantic section in a medical report document from which at least one slot in the prompt may potentially be filled according to the target schema.

We decide to judge the quality of an individual example using a combination of the F1 score and the accuracy for the relevant slot according to the evaluation results.

Notably, we decide to prefer examples with a higher F1 score in principle, but only if the example in question does not have a lower accuracy than the alternative example.

The reason why we decide to employ the accuracy as a sanity check stems from the fact that the F1 score will always be zero for any slot in the prompt for which no true positive results are found during the evaluation of the current choice of examples.

We therefore use the accuracy to determine which of two different examples to choose in cases where one example is clearly better than the other — with regards to the number of true negative, false positive and false negative evaluation results — but where both of the examples have an equal — and potentially even zero — F1 score.

4.5.2 Minification

During each minification step, the algorithm attempts to further improve the choice of examples by removing one of the examples chosen during the rotation phase.

The algorithm terminates once the only remaining examples are exactly those that the algorithm has determined must be kept during the previous minification steps.

The algorithm attempts to remove the example chosen for every slot in the prompt during the rotation phase exactly once, in the decreasing order of the best evaluation score seen for each slot across all previous evaluation steps during the rotation phase.

The algorithm subsequently evaluates both the previous choice of examples — which includes the example in question — and the choice of examples remaining after removing the example in question against documents from the *dev* split of the dataset.

The same evaluation procedure is used during the minification phase as during the rotation phase, except that all of the documents from *dev* split of the training dataset are always used during the minification phase instead of a subset of the documents.

Notably, we decide to prioritize more representative evaluation results over fewer total LLM inference requests during the minification phase, since the number of total minification steps is always bounded by the number of slots in the prompt.

4.6 Experiments

4.6.1 Training Setting

We perform prompt example pre-retrieval for a number of different prompts using an implementation of the prompt example tuning algorithm we describe in Section 4.5.

We manually design the individual prompts — each comprising a certain number of slots in the well-known target schema — such that the slots in a given prompt concern the extraction of similar types of information — such as pre-existing conditions or events occuring during hospitalization — and share the same slot value data type.

The aforementioned design method yields four distinct groups of prompts, where the slots in each of the groups are then further divided into multiple individual prompts.

We decide to divide the slots in a given group into multiple individual prompts, in the hope that designing each individual prompt such that it only comprises slots with a similar semantic meaning will improve the quality of the LLM inference results.

Further, we also make the decision to divide the slots in the target schema across multiple individual prompts due to that fact that the number of input tokens that may be accepted by an LLM is inherently limited. We argue that designing each individual prompt to comprise a low number of slots — and therefore a low number of total in-context examples — allows for example and query chunks of a larger size.

We refer to the source code repository[8] for the concrete definition of every single prompt we design according to the aforementioned method and experiment with.

We perform training for three different variants of each prompt in order to be able to demonstrate the efficacy of the prompt example tuning algorithm implementation.

The ZERO variant includes only the natural language instructions and the names of the slots in the given prompt — which are shared between every variant of the same prompt — but does not contain any in-context examples for the slots in the prompt.

The ZERO variant therefore represents a zero-shot scenario which we use to measure the baseline performance of the LLM for the common natural language instructions.

The BEST variant includes the choice of examples yielded by the rotation phase of the algorithm — that is, the locally optimal examples found for every slot in the prompt by performing a semi-random greedy search over the available examples.

The BEST variant therefore represents a few-shot scenario with the complete choice of original examples before removing any examples during the minification phase.

The LAST variant includes the choice of examples remaining after the minification phase of the algorithm — where zero or more of the original choice of examples may have been removed in an attempt to further improve the overall choice of examples.

The LAST variant therefore represents a few-shot scenario with the remaining choice of examples resulting from the removal of examples during the minification phase.

4.6.2 Prompt Template

We use the following template and natural language instructions — conforming to the syntax of the *jinja2* Python library — for all prompts during experimentation.

Notably, the template actually contains a typo — see "Use *the these* examples ..." — which was unfortunately only discovered after the conclusion of the experiments.

The template includes natural language instructions for teaching the LLM how to perform the domain-specific slot filling task, contains a list of the names for all the slots to fill and provides a number of suitable in-context examples.

[8] `https://github.com/semantic-systems/medical-slot-filling`

```
Extract the following information from the given text and answer with a valid JSON object containing only fields with the following keys and the corresponding extracted values:

{% for slot in slots -%}
- {{ slot }}
{% endfor %}

Use the these examples to understand how to generate an answer for a given text:

--- START OF EXAMPLES ---

{% for example in examples %}
Text: \"""
{{ example.text }}
\"""
Answer: {{ example.answer }}

{% endfor -%}

--- END OF EXAMPLES ---

Text: \"""
{{ context }}
\"""
Answer:
```

The concrete structure and natural language instructions used in the template are the result of manual experimentation and are specifically optimized to yield the best performance for the particular deployment configuration of the available LLM.

4.6.3 Evaluation

We evaluate the choice of examples obtained for every prompt and variant during training against the medical report documents from the *test* split of the dataset.

The complete evaluation results for the different prompts are included in Appendix A.

Table 2 shows the evaluation results for the first prompt group, which concerns the extraction of structured information relating to pre-existing conditions of the patient.

We can observe that the evaluations results between the BEST and LAST variant are identical for the **1-2** and **1-3** prompts, which indicates that no examples were removed during the minification phase. The final choice of examples after the completion of the minification phase is therefore identical to the locally optimal choice of examples yielded by the rotation phase, which explains the evaluation results for these prompts.

However, we can also observe that the evaluation results differ between the BEST and LAST variant for the **1-1** prompt, indicating that some of the examples yielded by the rotation phase were in fact removed during the minification phase in this case.

While the evaluation results shown in Table 2 for the **1-1** prompt appear to be worse for the LAST variant with regards to the overall recall, the overall F1 score and accuracy have clearly improved compared to the BEST variant of the **1-1** prompt.

We can further observe that the evaluation results for the LAST variant of the **1-4** prompt appear to be worse than for the BEST variant, which is especially noteworthy.

Normally, the evaluation results for the BEST variant constitute a lower bound for the evaluation of the BEST and LAST variants. However, the discrepancy between the evaluation results for the BEST and LAST variants in this specific case can be explained by a bug in the source code of the training algorithm implementation.

Table 3 shows the evaluation results for the second prompt group, which concerns the extraction of structured information relating to the resuscitation of the patient in question during or immediately prior to the admission of the patient to the hospital.

We can observe that no examples were removed during the minification phase for either of these two prompts. We can further observe that the evaluation results for the **2-2** prompt appear to be unusually poor, especially regarding the overall recall.

The bad recall can be explained by the fact that the **2-2** prompt is the first prompt that is comprised of slots that are not of the boolean value data type. The slots in the **2-2** prompt are instead of the text value data type.

We can thus begin to infer that the evaluation results for slots of the text value data type appear to be worse than the results for slots of the boolean value data type.

Table 4 shows the evaluation results for the third prompt group, which concerns the extraction of structured information relating to the reason the patient in question was admitted to the hospital and various measurements taken at the time of admission.

We can observe that no examples were removed either during the minification phase for any of the prompts in the third prompt group. We can further observe that the evaluation results for the **3-2** and **3-3** prompts — which are comprised of slots of the text value data type — are yet again worse compared to prompts comprising slots of the boolean value data type — such as the **3-1** prompt — continuing the trend.

Table 5 shows the evaluation results for the fourth prompt group, which concerns the extraction of structured information relating to the treatment given to the patient and events occurring during the stay of the patient in question at the hospital.

We can observe that some examples were removed during the minification phase for the **4-5** prompt, but not for any of the other prompts in the fourth prompt group.

While the removal of examples for the **4-5** prompt has resulted in an increase in the overall F1 score and accuracy, the overall accuracy and precision still appear to be unusually low considering that the **4-5** prompt is comprised of slots of the boolean value data type, compared to other similar prompts we have discussed previously.

This finding may potentially be explained by the fact that the number of slots in the **4-5** prompt is substantially higher than the number of slots in any other prompt.

The evaluation results of the **4-7** prompt appear to be rather poor as well, although this finding is less surprising considering that the slots of which the **4-7** prompt is comprised of are all of the text value data type. We can therefore observe a clear trend with regards to the effect of the slot value data type on the evaluation results.

The results from evaluating the choices of examples obtained during training against the *test* split of the dataset clearly indicate that we are capable of identifying true negative values across all the slots in the target schema with a high rate of success.

While the evaluation results also show a relatively high rate of false positive results, we deem this to be acceptable due to the specific nature of the problem at hand, as well as the implied constraints with regards to how we expect the implementation of the proposed system architecture to be deployed and used at the UKE in practice.

Prompt	**F1 score**	**Accuracy**	**Precision**	**Recall**
1-1$_{\text{ZERO}}$	0.15	0.75	0.57	0.09
1-1$_{\text{BEST}}$	0.63	0.73	0.52	0.94
1-1$_{\text{LAST}}$	0.76	0.82	0.70	0.89
1-2$_{\text{ZERO}}$	0.02	0.73	0.10	0.01
1-2$_{\text{BEST}}$	0.57	0.72	0.44	0.86
1-2$_{\text{LAST}}$	0.57	0.72	0.44	0.86
1-3$_{\text{ZERO}}$	0.00	0.81	0.00	0.00
1-3$_{\text{BEST}}$	0.39	0.61	0.28	0.78
1-3$_{\text{LAST}}$	0.39	0.61	0.28	0.78
1-4$_{\text{ZERO}}$	0.00	0.94	0.00	0.00
1-4$_{\text{BEST}}$	0.29	0.72	0.18	0.97
1-4$_{\text{LAST}}$	0.27	0.70	0.17	0.97

Table 2: Table showing the overall evaluation statistics for the first prompt group.

Prompt	**F1 score**	**Accuracy**	**Precision**	**Recall**
2-1$_{\text{ZERO}}$	0.31	0.83	0.38	0.33
2-1$_{\text{BEST}}$	0.54	0.72	0.39	0.89
2-1$_{\text{LAST}}$	0.54	0.72	0.39	0.89
2-2$_{\text{ZERO}}$	0.00	0.00	0.00	0.00
2-2$_{\text{BEST}}$	0.28	0.58	0.25	0.39
2-2$_{\text{LAST}}$	0.28	0.58	0.25	0.39

Table 3: Table showing the overall evaluation statistics for the second prompt group.

Prompt	**F1 score**	**Accuracy**	**Precision**	**Recall**
3-1$_{\text{ZERO}}$	0.00	0.78	0.00	0.00
3-1$_{\text{BEST}}$	0.48	0.58	0.34	0.81
3-1$_{\text{LAST}}$	0.48	0.58	0.34	0.81
3-2$_{\text{ZERO}}$	0.00	0.00	0.00	0.00
3-2$_{\text{BEST}}$	0.39	0.76	0.32	0.51
3-2$_{\text{LAST}}$	0.39	0.76	0.32	0.51
3-3$_{\text{ZERO}}$	0.00	0.19	0.00	0.00
3-3$_{\text{BEST}}$	0.26	0.46	0.46	0.26
3-3$_{\text{LAST}}$	0.26	0.46	0.46	0.26

Table 4: Table showing the overall evaluation statistics for the third prompt group.

Prompt	**F1 score**	**Accuracy**	**Precision**	**Recall**
4-1$_{\text{ZERO}}$	0.00	0.71	0.00	0.00
4-1$_{\text{BEST}}$	0.84	0.90	0.75	0.97
4-1$_{\text{LAST}}$	0.84	0.90	0.75	0.97
4-2$_{\text{ZERO}}$	0.32	0.85	0.45	0.28
4-2$_{\text{BEST}}$	0.59	0.82	0.47	0.92
4-2$_{\text{LAST}}$	0.59	0.82	0.47	0.92
4-4$_{\text{ZERO}}$	0.10	0.90	0.22	0.06
4-4$_{\text{BEST}}$	0.35	0.84	0.24	0.67
4-4$_{\text{LAST}}$	0.35	0.84	0.24	0.67
4-5$_{\text{ZERO}}$	0.00	0.91	0.00	0.00
4-5$_{\text{BEST}}$	0.35	0.72	0.24	0.91
4-5$_{\text{LAST}}$	0.38	0.76	0.26	0.91
4-6$_{\text{ZERO}}$	0.00	0.86	0.00	0.00
4-6$_{\text{BEST}}$	0.41	0.79	0.33	0.63
4-6$_{\text{LAST}}$	0.41	0.79	0.33	0.63
4-7$_{\text{ZERO}}$	0.00	0.05	0.00	0.00
4-7$_{\text{BEST}}$	0.34	0.65	0.24	0.62
4-7$_{\text{LAST}}$	0.34	0.65	0.24	0.62

Table 5: Table showing the overall evaluation statistics for the fourth prompt group.

Notably, we operate under the assumption that all results produced by the proposed system for medical report document analysis will have to be manually verified, seeing as the data quality standards posed by the research project in the context of which we solve the problem at hand necessitate that utmost care is taken to ensure that only accurate patient case data is inserted into the database maintained by the UKE.

We therefore deliberately choose to build a system that optimizes for a high recall, that is, produces as few false negative results as possible, at the cost of tolerating even a moderate to high number of false positive results. The implication of this choice is that the amount of human labor required for database maintenance at the UKE may be reduced, allowing personnel tasked with data entry to focus on double-checking positive results produced by the proposed system, rather than having to extract all of the relevant information from the document in question manually.

4.6.4 Limitations

While the previously discussed evaluation results generally show a promising overall recall across the different prompts, we believe the evaluation results may be further improved through a simple change in the creation process of the training dataset.

Notably, we believe that the available example chunks from which we choose the in-context examples throughout the experiments are not created in an optimal way.

We initially decided to use a rather low chunk size in an attempt to reduce the total number of input tokens for LLM inference requests. However, we believe that using a larger chunk size may have been beneficial in retrospect, since it is plausible that larger chunks may be more semantically meaningful to the LLM during inference.

We further did not initially use punctuation characters as chunking delimiters, which sometimes results in the text of the chunks being split up in the middle of a sentence.

Unfortunately, due the high average LLM inference response time, we were not able to consider different versions of the training dataset throughout the experiments.

We make another concession due to the high average LLM inference response time and manually set a hard limit of 10 iterations during the rotation phase, because unlike during the minification phase — where the number of iterations is bounded by the number of slots — the number of iterations during the rotation phase is bounded by the highest number of available examples for any slot in the prompt.

Therefore, we introduce an explicit limit for the number of iterations during the rotation phase in order to reduce the total time taken by the training experiments.

Because of the particular semi-random strategy we implement for selecting examples for a given slot, we still believe that the results we demonstrate are representative.

Further, we discovered a bug in the implementation of the training logic responsible for keeping track of the best example seen so far for a particular slot in the prompt.

Unfortunately, by the time the bug was discovered and fixed, there was a lack of sufficient time left and available computing resources. Therefore, we were unable to re-run the full training suite and obtain correct evaluation results for every prompt.

However, a preliminary analysis indicates that the bug in question only appears to have affected a single prompt, namely the **1-4** prompt. We therefore believe that the evaluation results we discuss for all of the other prompts are unaffected and correct.

The most important limitation that affects the experiments stems from the fact that the algorithm only allows for a maximum of one example to be chosen for every slot.

We deliberately design the algorithm with a top-down approach for example selection and removal in mind — rather than constructing the choice of in-context examples using a bottom-up approach — in order to avoid a brute-force search over the space of all combinations of available examples and severely reduce the number of decisions.

Accordingly, we decide to choose at most one in-context example per slot, which means that the algorithm is only ever required to make a binary decision concerning the choice of the example for a particular slot in the prompt during each iteration.

Notably, the algorithm may in theory still yield a choice of in-context examples with more than one total example for some slots in the prompt, as the example chosen for any particular slot in the prompt might be assigned more than a single slot name label. However, this is unlikely to occur in practice as we specifically implement the selection of examples to prefer example chunks with fewer slot name labels assigned.

We hypothesize that one possible reason for the comparatively poor performance with regards to slots of the text value data type might be that we provide the LLM with an insufficient number of in-context examples for this particular kind of slot.

We further completely omit all slots of the enumeration value data type in the well-known extraction target schema from the training experiments for a similar reason.

The problem with regards to slots of the enumeration value data type is that the "gold" slot answer corresponds to a pre-defined range of possible values, which is more resemblant to multi-label classification rather than to information extraction.

The text from which the answer for the majority of these slots may be determined however usually does not explicitly mention the desired answer value. The desired answer value is instead merely implied by the semantic meaning in the concrete text.

We thus face a challenge both when constructing the "gold" answer of an in-context example and when parsing the model answer during inference for these specific slots.

Finally, we also omit a small number of slots from the experiments that concern the extraction of sensitive patient information due an acute lack of available examples.

4.7 Discussion

RQ2 concerns the question of how we can employ the help of LLMs for the purposes of domain-specific structured information extraction from medical report documents.

We reduce the domain-specific structured information extraction problem at hand to a natural language comprehension challenge by framing it as a slot filling task.

The deliberate framing of the problem at hand as a slot filling task importantly enables us to employ the help of LLMs for structured information extraction by performing natural language comprehension over a given medical report document.

RQ3 concerns the question of how we can perform the domain-specific slot filling task by using in-context examples, in order to avoid the need to fine-tune an LLM.

Notably, taking advantage of the capabilities of LLMs for zero-shot natural language comprehension further allows us to avoid the need for task-specific model fine-tuning.

We instead teach the LLM how to perform the domain-specific slot filling task by making use of in-context learning and providing relevant examples during inference.

The evaluation results we discuss in Section 4.6.3 clearly demonstrate the efficacy of the approach we implement for addressing the second and third research question.

5 Summary

We describe and implement an architecture for a domain-specific document analysis system capable of extracting structured information from medical report documents.

We first fine-tune a multi-modal language model to recognize the structure of the visually-rich digital documents, which subsequently enables us to segment the text from each of the distinct semantic sections within a given medical report document.

We then frame the problem of extracting structured information from the section text segments as a slot filling task and employ in-context examples to perform inference.

Notably, we introduce a heuristic tagging approach using regular expressions to derive the ground truth labels for the section text segments we use as in-context examples.

We further propose a novel strategy for selecting suitable in-context examples based exclusively on the *a priori* knowledge about which slots may potentially be filled from each of the distinct semantic sections within a given medical report document.

Finally, we evaluate the implementation of the system architecture and interpret the results to demonstrate that we are able to successfully solve the problem at hand.

6 Future Work

We unfortunately faced several significant constraints due to the limited computing resources and time available for the purposes of our work. Notably, the average LLM inference response time was much longer than we originally anticipated. Accordingly, we were largely unable to experiment with different versions of the training dataset, or with different variations in the natural language instructions used for prompting.

Therefore, we suggest that future work focus especially on exploring whether the use of different chunking delimiters or a larger chunk size might further improve the evaluation results we demonstrate in Section 4.6.3. Further, we suggest that future work investigate whether optimizing the natural language instructions used for prompting might also lead to additional improvements in the evaluation results.

After performing a preliminary analysis of the evaluation results we demonstrate for the LAST variant of every prompt in Section 4.6.3, we find that the wildly varying and generally poor performance of the zero-shot baseline — especially with regards to the recall metric — is largely due to the inconsistent format of the LLM inference responses, which causes difficulties in parsing the slot answers predicted by the LLM.

We thus believe that researching how to better control the answer format of the LLM inference responses might be a particularly worthwhile topic for future works as well.

Slot	F1 score	Accuracy	Precision	Recall
Arterial hypertension	0.96	0.96	0.92	1.00
Diabetes mellitus	0.93	0.96	1.00	0.88
Hypercholesterinemia	0.85	0.91	0.73	1.00
Smoking status	0.58	0.64	0.46	0.79
COPD	0.50	0.87	0.43	0.60
Malignancies	1.00	1.00	1.00	1.00
Pacemaker/ICD/CRT	0.50	0.42	0.33	1.00

Table 6: Table showing the evaluation statistics for the 1-1$_{\text{LAST}}$ prompt.

Slot	F1 score	Accuracy	Precision	Recall
Coronary artery disease	0.66	0.53	0.50	0.95
Chronic kidney disease	0.73	0.73	0.59	0.94
Pulmonary arterial hypertension	0.24	0.71	0.15	0.50
Peripheral arterial disease	0.58	0.78	0.41	1.00
Thyreoid disease	0.67	0.82	0.53	0.89

Table 7: Table showing the evaluation statistics for the 1-2$_{\text{LAST}}$ prompt.

A Evaluation Results

The following tables show the complete evaluation results — including the different evaluation metrics for each individual slot — for the LAST variant of every prompt.

Note that we do not include evaluation results for the **4-3** prompt — which existed at an earlier point during the experiments — because the slots that the **4-3** prompt used to be comprised of have since either been integrated into the other prompts, or been completely omitted from the experiments due to difficulties during training.

We refer to the source code repository[9] for a list of the slots that have been omitted during the experiments, including the reasons why these slots have been omitted.

[9]https://github.com/semantic-systems/medical-slot-filling

Slot	F1 score	Accuracy	Precision	Recall
History of myocardial infarction	0.61	0.69	0.44	1.00
History of percutaneous coronary intervention	0.39	0.44	0.25	0.89
History of atrial fibrillation	0.83	0.82	0.71	1.00
History of stroke	0.20	0.29	0.11	0.80
History of congenital heart disease	0.00	0.93	0.00	0.00
History of cardiogenic shock	0.30	0.49	0.18	1.00

Table 8: Table showing the evaluation statistics for the 1-3_{LAST} prompt.

Slot	F1 score	Accuracy	Precision	Recall
History of CABG	0.56	0.82	0.42	0.83
History of aortic valve surgery/intervention	0.24	0.71	0.13	1.00
History of mitral valve surgery/intervention	0.22	0.36	0.12	1.00
History of tricuspid valve surgery/intervention	0.22	0.84	0.12	1.00
History of heart transplantation	0.29	0.89	0.17	1.00
History of left ventricular assist device	0.10	0.60	0.05	1.00

Table 9: Table showing the evaluation statistics for the 1-4_{LAST} prompt.

Slot	F1 score	Accuracy	Precision	Recall
Prior resuscitation	0.48	0.33	0.32	1.00
Bystander-witnessed cardiac arrest	0.40	0.80	0.27	0.75
Initial bystander-CPR	0.74	0.89	0.58	1.00
eCPR	0.53	0.84	0.40	0.80

Table 10: Table showing the evaluation statistics for the 2-1_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Cardiac rhythm at resuscitation	0.00	0.36	0.00	0.00
Cumulative duration of resuscitation	0.48	0.76	0.50	0.45
No-flow time	0.37	0.62	0.25	0.71

Table 11: Table showing the evaluation statistics for the 2-2_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Peripheral oedema	0.56	0.76	0.41	0.88
First diagnosis of HF?	0.40	0.40	0.27	0.75

Table 12: Table showing the evaluation statistics for the 3-1_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
NYHA state	0.28	0.53	0.24	0.33
INTERMACS level	0.00	0.60	0.00	0.00
Lactate at admission	0.67	0.89	0.62	0.71
pH at admission	0.25	0.87	0.17	0.50
QRS width	0.74	0.89	0.58	1.00

Table 13: Table showing the evaluation statistics for the 3-2_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Left ventricular ejection fraction	0.36	0.38	0.80	0.24
TAPSE	0.16	0.53	0.11	0.29

Table 14: Table showing the evaluation statistics for the 3-3_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Invasive ventilation	0.86	0.89	0.80	0.94
Non-invasive ventilation	0.82	0.91	0.69	1.00

Table 15: Table showing the evaluation statistics for the 4-1_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Dobutamine	0.62	0.87	0.50	0.83
Epinephrine	0.24	0.71	0.13	1.00
Norepinephrine	0.62	0.62	0.47	0.93
Milrinone	0.67	0.96	0.50	1.00
Levosimendane	0.80	0.93	0.75	0.86

Table 16: Table showing the evaluation statistics for the 4-2_{LAST} prompt.

Slot	F1 score	Accuracy	Precision	Recall
Treatment with MCS during hospital stay	0.55	0.60	0.38	1.00
LVAD implantation	0.00	0.96	0.00	0.00
Heart transplantation during hospitalization	0.50	0.96	0.33	1.00

Table 17: Table showing the evaluation statistics for the 4-4_{LAST} prompt.

Slot	F1 score	Accuracy	Precision	Recall
New-onset renal replacement therapy during hospital stay	0.53	0.64	0.36	1.00
CRT implantation	0.00	0.91	0.00	0.00
ICD implantation	0.22	0.84	0.12	1.00
Surgery/intervention for valvular disease	0.15	0.51	0.08	1.00
Coronary revascularisation	0.79	0.80	0.65	1.00
Right heart catheterization	0.40	0.80	0.25	1.00
MRI	0.33	0.64	0.20	1.00
Electrical cardioversion	0.34	0.49	0.21	1.00
Ablation therapy	0.57	0.93	0.40	1.00
Peripheral ischemia requiring an intervention	0.18	0.80	0.10	1.00
Abdominal compartment or mesenteric ischemia with need for laparotomy	0.67	0.96	0.50	1.00

Table 18: Table showing the evaluation statistics for the 4-5_{LAST} prompt.

Slot	F1 score	Accuracy	Precision	Recall
Resuscitation during hospitalisation	0.52	0.76	0.38	0.86
Death during hospital stay	0.86	0.89	0.76	1.00
Sepsis during hospital stay	0.76	0.84	0.65	0.92
Stroke	0.00	0.62	0.00	0.00
TIA	0.00	1.00	0.00	0.00
Bleeding with need for intervention or transfusion	0.33	0.64	0.20	1.00

Table 19: Table showing the evaluation statistics for the 4-6_{LAST} prompt.

Slot	**F1 score**	**Accuracy**	**Precision**	**Recall**
Date of first MCS device implantation	0.35	0.51	0.25	0.60
Date of last MCS device explantation	0.44	0.89	0.33	0.67
Type of Impella device	0.23	0.56	0.14	0.60

Table 20: Table showing the evaluation statistics for the 4-7_{LAST} prompt.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

[2] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712, 2023. doi: 10.48550/ARXIV.2303.12712. URL `https://doi.org/10.48550/arXiv.2303.12712`.

[3] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`. Accessed: 2024-01-29.

[4] Bernardo Cuteri. Closed domain question answering for cultural heritage. In Viviana Mascardi and Ilaria Torre, editors, *Proceedings of the Doctoral Consortium of AI*IA 2016 co-located with the 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016), Genova, Italy, November 29, 2016*, volume 1769 of *CEUR Workshop Proceedings*, pages 17–22. CEUR-WS.org, 2016. URL `https://ceur-ws.org/Vol-1769/paper03.pdf`.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[6] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. Robust retrieval augmented generation for zero-shot slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1939–1949, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.148. URL `https://aclanthology.org/2021.emnlp-main.148`.

[7] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document AI with unified text and image masking. In João Magalhães, Alberto Del Bimbo, Shin'ichi Satoh, Nicu Sebe, Xavier Alameda-Pineda, Qin Jin, Vincent Oria, and Laura Toni, editors, *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022*, pages 4083–4091. ACM, 2022. doi: 10.1145/3503161.3548112. URL `https://doi.org/10.1145/3503161.3548112`.

[8] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL `https://aclanthology.org/2022.deelio-1.10`.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

[10] Yannis Papanikolaou, Marlene Staib, Justin Joshua Grace, and Francine Bennett. Slot filling for biomedical information extraction. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 82–90, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bionlp-1.7. URL `https://aclanthology.org/2022.bionlp-1.7`.

[11] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a

benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.200. URL `https://aclanthology.org/2021.naacl-main.200`.

[12] Raphael Scheible, Fabian Thomczyk, Patric Tippmann, Victor Jaravine, and Martin Boeker. Gottbert: a pure german language model. *CoRR*, abs/2012.02110, 2020. URL `https://arxiv.org/abs/2012.02110`.

[13] R. Smith. An overview of the tesseract OCR engine. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*, pages 629–633. IEEE Computer Society, 2007. doi: 10.1109/ICDAR.2007.4376991. URL `https://doi.org/10.1109/ICDAR.2007.4376991`.

[14] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2022. URL `https://github.com/heartexlabs/label-studio`. Open source software available from https://github.com/heartexlabs/label-studio. Accessed: 2024-01-29.

[15] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971. URL `https://doi.org/10.48550/arXiv.2302.13971`.

[16] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kam-

badur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL `https://doi.org/10.48550/arXiv.2307.09288`.

[17] Jiapeng Wang, Lianwen Jin, and Kai Ding. LiLT: A simple yet effective language-independent layout transformer for structured document understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7747–7757, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.534. URL `https://aclanthology.org/2022.acl-long.534`.

[18] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.201. URL `https://aclanthology.org/2021.acl-long.201`.

[19] 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 Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1192–1200. ACM, 2020. URL `https://dl.acm.org/doi/10.1145/3394486.3403172`.

[20] Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florêncio, Cha Zhang, and Furu Wei. Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding. *CoRR*, abs/2104.08836, 2021. URL `https://arxiv.org/abs/2104.08836`.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Unterschrift [illegible] Ort, Datum Buchholz in der Nordheide, den 5. Februar 2024