

Article

NERFlow: A Workflow-Based Subsystem of FIT4NER for LLM-Assisted Medical Named Entity Recognition

Florian Freund , Philippe Tamla , Bao Tran and Matthias Hemmje 

Faculty of Mathematics and Computer Science, University of Hagen, 58097 Hagen, Germany; matthias.hemmje@fernuni-hagen.de (M.H.)

* Correspondence: florian.freund@fernuni-hagen.de

Abstract

Preparing training data for domain-specific medical Named Entity Recognition (NER) involves a trade-off between annotation quality, expert effort, and data privacy: manual annotation is costly, whereas cloud-based Large Language Models (LLMs) raise concerns about the control of sensitive clinical text. This article introduces NERFlow, a workflow-driven subsystem of the FIT4NER project whose contribution is an abstraction layer that makes rule-based, model-based, and LLM-based annotation interchangeable and comparable within one configurable workflow environment. Open-source LLMs are integrated as exchangeable annotation services, deployable locally or in cloud-agnostic infrastructures via Kubernetes, and embedded into the KM-EP knowledge management system. NERFlow was evaluated qualitatively, through a cognitive walkthrough, an IEEE 1028 technical review, and a user-centered survey with 18 participants, and quantitatively on the CRAFT corpus with seven open-source and hosted LLMs run through an identical pipeline. The results support its use as LLM-assisted pre-annotation with expert correction, with locally deployable open-source models as the more reliable basis for reproducible operation.

Keywords: natural language processing; named entity recognition; machine learning; large language models; medical expert systems; clinical decision support

1. Introduction and Motivation

Large Language Models (LLMs) have revolutionized text analysis methods, particularly in the area of medical **Named Entity Recognition (NER)** [1]. NER is a text analysis method in **Natural Language Processing (NLP)** used to extract knowledge from natural language texts, thereby reducing **Information Overload (IO)** [2]. In the medical field, **Evidence-based Medicine (EBM)** is used to develop **Clinical Practice Guidelines (CPGs)**, which translate the best available evidence into structured recommendations that assist healthcare professionals in decision-making and lower the risks of treatment for patients [2]. The development of such guidelines relies on extensive research and careful documentation of evidence [3]. This requires analyzing large amounts of unstructured text, including scientific literature, research findings, and clinical studies. NER facilitates the extraction of knowledge from these unstructured data sources and converts it into structured information according to Kuhlen's information model [2,4]. In this model, information emerges when signs are selected, structured, and contextualized in a way that reduces complexity for a specific cognitive task, for example, when a raw clinical sentence such as “*The patient was treated with amoxicillin for acute sinusitis*” is transformed into structured entities like [DIAGNOSIS: acute sinusitis] and [TREATMENT: amoxicillin], thereby enabling its systematic



Academic Editors: Andrzej Zbrzezny,
Toktam Ghafarian Mabhoot and
Tomasz Krzywicki

Received: 24 June 2026

Revised: 26 July 2026

Accepted: 5 August 2026

Published: 6 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

interpretation and reuse. NER directly supports this transformation by mapping raw text to semantically typed entities (e.g., diagnoses, findings, treatments), thereby providing the structured building blocks required to aggregate, interpret, and reuse medical evidence in CPGs. In medicine, specialized vocabularies are used to precisely define diagnoses, findings, and treatments [5]. Therefore, it is crucial for medical NER to accurately recognize domain-specific **Named Entities (NEs)**, as this directly affects the quality of downstream tasks such as evidence synthesis, decision support, and guideline development. Recent advances in **Artificial Intelligence (AI)** have led to the development of LLMs trained on very large data sets, elevating machine processing of text to a new level [6]. There are already approaches to using LLMs for NER without extensive annotated training data [1]. However, the accuracy of the results for medical applications remains inadequate [1], and there are concerns about data privacy, information security, and transparency when relying on commercial, cloud-based LLMs [7]. These models typically operate as remote services with limited control over data flows and model behavior, which complicates regulatory compliance and makes it difficult to trace or explain individual outputs. These limitations motivate the exploration of open-source LLMs that can be deployed locally or within cloud-agnostic infrastructures. Such models offer greater control over deployment, the possibility to restrict processing to trusted environments, and improved transparency regarding model architecture and, in some cases, training data, which is essential for sensitive medical **Use Cases (UCs)**.

Following this overview of context and terminology, the reasoning behind this study is motivated by relevant research initiatives. The European project **Recommendation Rationalisation (RecomRatio)** [8] supports healthcare workers by providing treatment options complemented by their pros and cons, assisting in informed decision-making. It relies on **Content and Knowledge Management Ecosystem Portal (KM-EP)**, a knowledge management system that supports evidence-based decisions sourced from the medical literature. Co-developed by the Chair of Multimedia and Internet Applications at the Faculty of Mathematics and Computer Science, FernUniversität in Hagen, along with FTK e.V. Research Institute for Telecommunications and Cooperation [9], KM-EP incorporates emerging NEs into medical documentation [10] in order to represent clinically relevant concepts in a structured way and make them available for search, analysis, and guideline development. As part of the proposal for the H2020 project **Artificial Intelligence for Hospitals, Healthcare & Humanity (AI4H3)**, a hub-centric architecture was recommended [11], designed to manage medical records and aid clinical decision-making. This architecture features a central AI hub, equipped with a Semantic Fusion Layer for model and data storage. AI4H3 integrates KlinSH-EP, an evolution of the KM-EP system [12]. Under the AI4H3 framework, the **Cloud-based Information Extraction (CIE)** project [13] was launched to build a cloud-based NER pipeline capable of processing large data sets. The CIE architecture provides shared infrastructure for configuring cloud resources and training NER models at scale, enabling multiple users and projects to reuse common services instead of setting up isolated pipelines. Building on this, the **Framework-Independent Toolkit for Named Entity Recognition (FIT4NER)** project [14], as part of the CIE initiative, focuses on making medical text analysis more flexible by supporting various NER frameworks and models. FIT4NER serves as a framework-independent toolkit built on CIE: it offers medical experts a testbed in which different NER approaches can be deployed, evaluated, and compared under realistic, cloud-backed conditions [13,14]. Within this setup, KM-EP remains the central **Knowledge Management System (KMS)** governing documents, such as clinical and research reports, which provide the evidence base for CPGs. To increase accessibility to these documents, KM-EP offers classification via NER and a faceted search engine based on the extracted entities [15]. The integrated **Machine Learning (ML)**-based NER models

in KM-EP should ideally be trained by medical professionals to handle domain-specific terminology, abbreviations, and potential spelling variations [5].

However, training custom ML-based NER models requires large amounts of annotated training data. Annotating extensive datasets is often laborious and time-consuming. The use of LLMs for automated annotation could reduce these efforts, but it also presents several challenges. First, the accuracy of LLMs for medical NER is still insufficient compared to systems trained on adequate data [1]. LLMs may generate inaccurate or fabricated information that sounds plausible but is not factually correct (hallucinations) and sometimes struggle to precisely determine the positions of entities within the text [1]. Second, widely used LLMs, such as GPT-5 or Gemini, are typically offered as cloud-based services and often do not disclose the data used for training, raising concerns about information security, data privacy, traceability, and explainability in the context of medical data [7,16,17]. In contrast, open-source LLMs allow full transparency regarding model weights and, in some cases, training data and can be deployed locally or in controlled cloud environments. Third, training data preparation and NER model training require standardized preliminary steps, such as data cleaning, annotation, and data splitting, which should be supported by flexible workflows and scalable computing resources.

Existing research typically addresses individual aspects, such as isolated annotation tools or project-specific pipelines [1,18,19], or focuses on general-purpose workflow and orchestration technologies that are not tailored to integrated, reusable preprocessing pipelines for medical NER [20–22]. As a result, existing solutions often lack unified workflow support capable of orchestrating the required preliminary steps across different NER frameworks and infrastructures in a cloud-agnostic manner.

The contribution of this work, therefore, does not lie in a single annotation method but in an architecture that makes such methods interchangeable. It provides an abstraction layer through which rule-based, model-based, and LLM-based annotation can be deployed, configured, and compared in terms of their results within the same workflow-based environment, without requiring any modification of the system. For LLM-based annotation, the same holds for switching between different open-source models. The automated annotation of medical texts using open-source LLMs is one method provided through this layer, which medical experts review and refine. The preliminary steps are standardized as workflows and integrated into KM-EP. Within FIT4NER, the same workflow- and plugin-based architecture supports not only annotation but the entire training process, from data preparation through training to the deployment of the models. The present contribution deliberately focuses on the workflow system and the creation of annotated training data, whereas the remaining process steps are part of the surrounding project rather than of this work. The quality that any individual model achieves in this annotation is therefore not a claim of this work but the subject of its empirical investigation in Section 5. Precisely because the architecture supports the interchange of methods, its usefulness can be assessed independently of how well a particular model annotates at any given point in time.

This leads to the following **Research Questions (RQs)**, addressed in a master's thesis inspired by the CIE and FIT4NER projects [23] and aligned with the **Remaining Challenges (RCs)** identified in the state of the art (Section 2):

- **RQ1:** How can a system with an integrated abstraction layer be developed that provides multiple methods for the automated creation of annotations using various open-source LLMs, ensures compatibility with the selected NER frameworks for training, and supports standardized preliminary steps for NER model training on cloud-agnostic infrastructures?
- **RQ2:** How can a system be developed that manages different annotation services, orchestrates them within workflow-based preliminary steps (data cleaning, annota-

tion, and data splitting), and allows users to configure these services with various parameters while integrating seamlessly into the KM-EP environment for medical knowledge management?

To address these questions, this study employs the Nunamaker methodology [24]. This well-established and structured framework delineates four distinct phases: **Observation**, **Theory Building**, **System Development**, and **Experimentation**. Section 2 explores observational objectives through an exhaustive review of related work and derives the RCs relevant to this work. Section 3 is devoted to the theory construction process, in which models are developed to assist medical experts in creating an annotated training corpus and to formalize workflow-based preliminary steps on cloud-agnostic infrastructures. The development of the system is described in Section 4, where the prototype for automated annotation using open-source LLMs, **Kubernetes (K8S)**, and workflow orchestration is introduced and integrated into KM-EP. The experimentation, which incorporates a quantitative assessment of the prototype, is examined in Section 5. Finally, Section 6 summarizes the main findings of the study and reflects on the identified RCs.

2. Related Work

This section corresponds to the observation phase of the Nunamaker methodology. It establishes the scientific and technological foundation for this study and related research initiatives. The section evaluates the current state of the art and identifies the RCs within the relevant fields. It begins by introducing NER and LLMs and by performing a literature review to identify existing projects that combine NER with LLMs. Subsequently, it examines the use of cloud technologies and workflow systems for medical NER model training. Finally, the section assesses the suitability of KM-EP as the target environment for integrating workflow-based preliminary steps and open-source LLMs in support of medical experts.

NER is a crucial NLP technique designed to extract NEs from unstructured text documents [25]. In the medical domain, NER plays a pivotal role in **Clinical Decision Support Systems (CDSSs)** and facilitates clinical information mining from **Electronic Health Records (EHRs)** [26]. Recent years have witnessed significant advances in NER due to the emergence of new techniques and models, particularly **Deep Learning (DL)** [6]. These innovations have led to substantial improvements in the performance of NER systems, establishing NER as one of the most researched tasks in NLP today [6]. Furthermore, pre-training LLMs, such as BERT [27], GPT-2 [28], or RoBERTa [29], on large corpora has demonstrated remarkable improvements in NER performance. These models achieve state-of-the-art results on NER tasks and can be efficiently fine-tuned on smaller datasets for domain-specific applications.

Beyond specialized transformer models, recent approaches leverage popular LLMs, often used as chatbots [30], for NER to further reduce the training effort. Various technologies are relevant for interacting with LLMs or improving their capabilities. **Prompting** refers to the technique of providing the model with specific hints or instructions (prompts) to elicit a desired response or output [31]. In the context of LLMs, this means that a user provides the model with text input that prompts it to perform a specific task or generate a particular type of response. Effective prompting can significantly improve the quality and relevance of the generated answers by clarifying the context and helping the model focus its capabilities. **Few-Shot Prompting** involves presenting the model with only a few examples for a specific task, allowing it to perform similar tasks accurately [31]. This method enhances the model's performance without necessitating an extensive training phase. In contrast, **Zero-Shot Prompting** requires the model to complete a task without receiving specific examples in advance [31]. The model relies on its general knowledge

and training data to perform the task, which is particularly useful when annotated data for a specific application are unavailable. **Fine-tuning** refers to the process of adapting a pre-trained model to specific data or tasks [27]. This is done to optimize the model's performance in a particular application area by training it on the unique characteristics and requirements of the dataset. Fine-tuning allows for the effective utilization and adjustment of the model's existing knowledge but requires sufficient training data.

Several studies investigate the use of LLMs to support or replace traditional NER approaches. Wang et al. [18] developed GPT-NER, a NER method that transforms sequence labeling into a text generation task by utilizing special tokens to mark entities. The method also incorporates a self-verification strategy to reduce hallucinations, i.e., instances where the model generates inaccurate or fabricated information that sounds plausible but is not factually correct. GPT-3 is used as the underlying model, employing In-Context Learning and Few-Shot Prompting. GPT-NER achieves performance comparable to fully supervised training methods and significantly exceeds them in scenarios with limited training data. The results demonstrate that the method is particularly effective in Few-Shot Prompting contexts. The work by Zeghidi and Moncla [19] also explores Few-Shot Prompting with LLMs for NER. This study employs models such as GPT-3.5, GPT-4, GPT-4o, as well as smaller models such as Phi3, Gemma, Mistral, Qwen, and Llama. NER is applied to the GeoEDdA dataset, which contains geographical and semantic annotations from the Diderot Encyclopedia. The findings indicate that LLMs, especially GPT-4o, achieve promising results in Few-Shot Prompting; however, they still lag behind specialized transformer models like **Bidirectional Encoder Representations from Transformers (BERT)**. A significant challenge lies in accurately determining the positions of entities within the text. LLMs often struggle to pinpoint these precisely, resulting in occasionally incorrect or hallucinated positions. Lu et al. [1] investigated the capabilities of LLMs in NER within clinical texts, focusing specifically on rare diseases, using Zero-Shot Prompting, Few-Shot Prompting, **Retrieval-Augmented Generation (RAG)**, and fine-tuning. Their analysis evaluates both proprietary models like GPT-4 and local open-source models such as Llama 2. The application focuses on extracting information about rare diseases and their phenotypes from the RareDis-v1 dataset. The results reveal that LLMs, particularly without fine-tuning, face challenges in NER, while Llama 2, after fine-tuning, outperforms GPT-4 and nearly matches the performance of BioClinicalBERT. Again, Few-Shot Prompting significantly enhances performance, while RAG offers limited advantages. These studies highlight that Few-Shot Prompting can significantly boost the NER performance of LLMs, particularly in scenarios with limited training data. In contrast, specifically trained transformer models like BERT generally exhibit higher performance when sufficient training data are available. Therefore, it is evident that, at present, training or fine-tuning customized NER models is necessary for specific medical use cases. However, LLMs remain promising tools for reducing the effort required to prepare training data for model training.

The creation of the training data itself has so far been carried out predominantly in dedicated annotation tools that represent different lines of development. *brat* [32] stands for classical, purely manual annotation in the browser. It supports nested and overlapping entities as well as relations between them and is regarded as the de facto standard for corpus creation without machine support. *INCEpTION* [33] augments manual annotation with recommendation services and active learning and integrates knowledge bases so that annotations can be tied to existing vocabularies. *MedTAG* [34] is tailored to biomedical documents, offering customizable annotation schemas and a low installation overhead. *Prodigy* [35] represents the commercial branch. It is designed as a scriptable developer tool and places active learning at its center. These tools share two properties that are decisive for the UC considered here. First, each of them commits to a single annotation paradigm.

Where machine support is provided, it originates from one fixed source: a recommendation service, a statistical model, or a learning procedure. None of the tools makes it possible to run rule-based, model-based, and LLM-based procedures side by side on the same texts and then compare their results. Domain experts who, without prior NLP knowledge, are asked to judge which procedure is suitable for their corpus must decide on a procedure before annotation, even though it could only be assessed afterward. Second, none of the tools are embedded in a KMS in which the documents to be annotated are already managed. The creation of training data remains a separate step in a separate tool, with its own import, its own user management, and its own export. Both observations support the need for a flexible annotation solution that is integrated into a KMS.

The tools mentioned above must be distinguished from approaches that bind annotations directly to a controlled vocabulary. Knowtator [36], an extension of the Protégé platform, uses the ontologies defined there directly as the annotation schema. It was used to annotate the **Colorado Richly Annotated Full Text (CRAFT)** corpus [37], which serves as the reference corpus in this work. This touches on a distinction that is frequently blurred in the literature: mapping text spans to concepts of an ontology is not NER with a small label set but concept normalization. It combines the recognition of a mention with its normalization to a controlled vocabulary, such as the UMLS [38]. cTAKES [39] is the long-standing reference system, and more recent work by Schäfer et al. [40] combines recognition, linking, and normalization in a single processing pipeline. For the exchange between tools and processing pipelines, BioC [41] has become established. It is a deliberately minimal format that captures distributed, nested, and overlapping annotations as well as relations between them, and it can be integrated into existing NER systems with little effort.

To reduce the manual effort, the source text is usually pre-annotated automatically and subsequently reviewed by domain experts. That this approach is effective was demonstrated by Lingren et al. [42] for disease and symptom entities in clinical trial announcements: the time savings were statistically significant across all experimental settings, without the pre-annotation biasing the results. This finding, however, concerns *dictionary-based* pre-annotation, whose errors follow from the dictionary coverage and remain recognizable to annotators. Whether it carries over to LLMs, whose outputs are phrased plausibly and are therefore harder to refute, is an open question. For LLMs as an annotation tool, the instructive part of the finding by Ding et al. [43] lies in the cost perspective: under a *fixed budget*, a model trained on GPT-3-annotated data often matches or exceeds one trained on human annotations, because the same amount of money buys substantially more annotated data. On a per-annotation basis, the quality trails behind, but the advantage arises from volume rather than from accuracy. That study considers only GPT-3 and explicitly not open-source models, where computing time and model quality tend to be the limiting factors instead. How strongly the design of the prompt determines the quality was measured by Hu et al. [44] for two clinical NER tasks on GPT-3.5 and GPT-4. Their prompt scaffold, consisting of a task description, annotation guidelines, correction hints from an error analysis, and annotated examples, raises the recognition performance so far that GPT-3.5 with the full scaffold outperforms GPT-4 with only a task description. The prompt thus weighs more heavily than the leap between two model generations. Even the best prompt configuration still trails the purpose-trained BioClinicalBERT model, although the latter requires substantial training data.

LLMs are often offered as cloud-based services, such as ChatGPT from OpenAI, Grok from xAI, or Gemini from Google. This raises concerns about information security and data privacy, particularly in the context of medical data [7,16]. Additionally, widely used LLMs, such as GPT-5 and Gemini, often do not disclose the data used for training, leading to issues related to traceability and explainability of the results [17]. In contrast to these

commercial LLMs, there are open-source models such as Llama, DeepSeek, Phi, or Mistral [45], which offer permissive licenses and, in some cases, publish their training data or at least their model weights [46]. These models can be self-hosted on open-source platforms, providing improved transparency and control over data flows, which is crucial for medical applications. Although this approach mitigates data privacy concerns associated with cloud-based commercial LLMs, running LLMs locally requires substantial computing resources, and these open-source models are generally less accurate than their commercial counterparts [46]. **This analysis leads to RC1: *Medical experts should be supported by open-source LLMs in the development of training data for the training of NER models.***

That data privacy and licensing questions actually govern tool selection in practice is empirically documented. In a survey of NER tool users, both are repeatedly named as major obstacles [47]. Running models locally, however, shifts the risk rather than removing it. Zhao et al. [48] demonstrate backdoor attacks in which the trigger does not lie in inserted words but in the writing style of the training examples, and in which the examples remain correctly labeled. Both of the features by which poisoned data can otherwise be recognized thus disappear. The attacks achieve almost complete success. For the approach pursued here, this entails a limitation of the review step: a domain review that confirms the labels does not yet establish that the resulting corpus is harmless. The provenance of the models and of the texts remains a separate security question.

The model training process to support the development of CPGs includes a phase of “Data Management and Curation”, which involves various preliminary steps for model training [2]. These steps consist of data cleaning, data annotation, and data splitting into training and test sets. One way to leverage LLMs in this process is to automate the annotation of medical texts. Annotation refers to the addition of markers or labels to words, phrases, or sentences in a text to highlight specific features or information. Workflows can be used to perform these preliminary steps, including automated annotation. A workflow is a sequence of activities designed to accomplish a specific task [49]. Workflows are applicable in both scientific and business contexts, automating processes where various logical data processing activities handle data according to established rules. To execute a workflow, a workflow engine is required, which links and processes the workflow specification with the provided information. The typical components of a workflow are activities (tasks and steps), such as data processing, file uploads, or model training [49]. These activities can be executed sequentially or in parallel. Workflow data, which include input data, intermediate results, and output data, can be in various formats, such as XML or JSON. The structure of a workflow is often represented as a **Directed Acyclic Graph (DAG)**, indicating that a workflow is finite and triggered by an event (e.g., HTTP call) and completed with the supplied data. In addition, there are various control flow constructs, such as conditions, loops, and parallel processing. Executing workflows requires computing power, often provided in the cloud. Workflow monitoring typically involves logging runtime metrics, errors, and the status of workflow executions. Workflows provide a flexible means to structure preliminary steps for the training of NER models within defined process flows. Different workflows can be established, allowing for variations based on the specific LLMs employed. Individual process steps can easily be swapped out, which is beneficial given ongoing advances in AI and NER. Ultimately, NER experts can design and define these workflows, facilitating the preliminary steps for medical professionals and contributing to standardized and reproducible processes. **This leads to RC2: *The preliminary steps (data cleaning, data annotation, and data splitting) for medical NER model training should be supported and standardized through the use of workflows.***

Both training NER models and using workflows with steps based on LLMs require computing resources that may not be available locally or may only be needed temporarily.

The use of cloud technologies provides a solution, as the necessary computing resources do not need to be maintained on-site and can be utilized as needed. There are various **Cloud Service Providers (CSPs)** that offer a wide range of products and services, such as **Amazon Web Services (AWS)**, Microsoft Azure, or **Google Cloud Platform (GCP)**. This creates a significant risk of vendor lock-in, meaning that integration of an environment with a chosen CSP can become so extensive that switching to another provider would be prohibitively expensive and complex [50]. Furthermore, portability of the environment to other CSPs is hindered [50]. **Cloud-agnostic** refers to an architecture or system designed to operate on different cloud platforms or infrastructures without being tied to a specific CSP [51,52]. This means that applications or services in a cloud-agnostic environment are not bound to a particular cloud technology and can easily migrate between various providers or even to on-premises infrastructures [51,52]. Cloud-agnostic systems often rely on open standards and platform-independent technologies, such as containerization (e.g., Docker) and orchestration systems [51,52]. This allows applications to be developed and operated in a way that functions consistently across different cloud environments. Despite their advantages, such as portability, resilience, and strategic flexibility, cloud-agnostic architectures also present significant challenges. In particular, more effort is required for development and maintenance, and there may be limited options for utilizing specific cloud services. K8S [53] is an open-source cloud-agnostic system designed to automate the deployment, scaling, and management of containerized applications. Its primary goal is to handle common tasks associated with the deployment of applications in the form of containerized microservices. These tasks include starting, stopping, or restarting applications, monitoring load and potentially launching additional application instances, routing network traffic to various application instances, and other related functions. K8S can be installed on a cluster composed of multiple machines or **Virtual Machines (VMs)**, or it can be offered as a service by CSPs such as AWS, GCP, or Azure. Empirical studies demonstrate that K8S consistently outperforms alternative orchestration engines such as Docker Swarm, Apache Mesos, and Cattle, particularly in the context of complex application deployments [22]. Although lighter engines may offer advantages for simple UCs, K8S stands out as one of the most feature-rich and comprehensive orchestrators available, which explains its widespread adoption in practice [22]. Therefore, in the context of medical data and NER model training, it is recommended to use cloud-agnostic technologies such as K8S to flexibly and dynamically access computing resources from public clouds, specialized medical clouds, or local environments. **This leads to RC3: K8S should be utilized as a cloud-agnostic layer to support the execution of preliminary steps for NER model training.**

KM-EP has been utilized multiple times as a KMS in the medical field [12,54]. The system enables efficient management of medical research findings and provides them as a foundation for NER model training by medical experts. It offers a unified user interface and supports essential functions such as user and rights management. Combined with the modern graphical user interface and the locally available medical research findings, it is logical to expand KM-EP with features for NER model training and the necessary preparatory tasks using a workflow system. There are several workflow systems available. Apache Airflow [20] is a scheduler for batch and **Extract-Transform-Load (ETL)** workloads based on DAGs. It benefits from a large ecosystem but is less suitable for event-driven, long-running, or stateful processes. Temporal [21] provides strong guarantees for durable, stateful workflows with exactly-once semantics but comes with a complex operational stack. Inngest [55] is a serverless, event-driven workflow platform that simplifies asynchronous processes such as background jobs and multi-step workflows. It combines events, queues, and persistent execution, allowing developers to focus on business logic. For this work, Inngest offers clear advantages: explicit event modeling, automatic retries and idempotency,

strong developer ergonomics, and easy integration with containerized jobs. In contrast, Temporal and Airflow are operationally heavier or less suited for fine-grained, stateful, and event-driven training workflows. **This leads to RC4: An Inngest-based workflow function for the preparatory tasks involved in training ML-based NER models, including automated annotation based on LLMs, should be integrated into KM-EP.**

In this section, a critical analysis of the current state of the art relevant to this research has been carried out, leading to the identification of four RCs. The first challenge (RC1) involves the use of open-source LLMs for the creation of training data for ML-based NER. The second challenge (RC2) focuses on workflows for a standardized process for training data creation through medical professionals. The third challenge (RC3) concerns the implementation of K8S as a cloud-agnostic computation layer supporting the preliminary steps of NER model training. Finally, RC4 involves the selection of KM-EP as the target environment and its extension with an Inngest-based workflow function for the preparatory tasks involved in training ML-based NER models, including automated annotation based on open-source LLMs. The subsequent section will provide a comprehensive account of the development of appropriate models designed to tackle all identified challenges (RC1–4).

3. Modeling

Following the analysis of the current state of the art and the identification of the RCs in Section 2, this section advances to the theory-building phase of the Nunamaker methodology. The goal is to develop a set of conceptual models for a subsystem that enables medical experts to efficiently create, review, and refine training data for ML-based NER models across different NER frameworks. These models form the conceptual foundation of the **NERFlow** subsystem, developed within the FIT4NER project and based on a master's thesis [23].

The name *NERFlow* reflects the system's core purpose: it provides a workflow-driven ("Flow") environment in which preprocessing and annotation steps can be orchestrated, executed, and evaluated in a systematic and reproducible manner. By combining configurable workflows, microservice-based processing, and open-source LLMs, NERFlow establishes an end-to-end flow of data from raw text to expert-validated training corpora. NERFlow is explicitly designed to address four key challenges identified in the state of the art: (1) leveraging open-source LLMs for annotation assistance (RC1), (2) standardizing preliminary steps such as cleaning, annotation, and data splitting through workflows (RC2), (3) enabling cloud-agnostic execution via K8S (RC3), and (4) integrating these capabilities into KM-EP as the target environment for medical knowledge management (RC4). To structure user interactions and system behavior, the **User Centered System Design (UCSD)** method [56] is applied, and **Unified Modeling Language (UML)** [57] is used for formal specification.

Data-driven machine learning applications require high-quality training datasets. To support medical experts in creating such datasets, FIT4NER defines the use cases "Extract Data" and "Support Data Preparation" [14]. The latter comprises three core activities: "Clean Data" (removing noise), "Annotate Data" (assigning labels), and "Visualize Data" (providing corpus overviews).

These core UCs, highlighted in green in Figure 1, are performed by **Model Definition Users** and are extended in NERFlow to address RC1 and RC2 by supporting automated preprocessing through configurable workflows. Extended UCs include "Execute Annotation Workflow," which processes data through sequential or parallel pipelines. Each workflow step performs extraction, cleaning, transformation, or annotation using open-source LLMs. Workflow outputs are visualized in the "View Annotated Data" UC for correctness checks, while "Process Annotated Data" allows users to refine or correct annotations. "Compare

“Annotated Data” supports document comparison and consolidation of annotations for training datasets. Reusable workflow templates are not created by the medical experts themselves but by users with administrator rights, who create, edit, and exchange them in order to further reduce the training effort required of the experts.

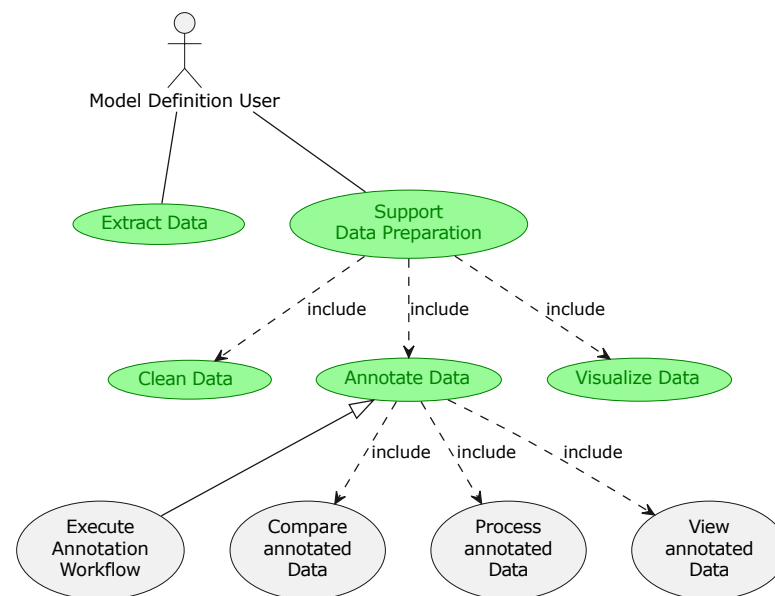


Figure 1. Use context and UML use cases for Model Definition Users.

Figure 2 shows the structural composition of NERFlow together with its embedding into the surrounding FIT4NER architecture. Within the NERFlow component, the *workflow engine* acts as the central execution instance, together with the building blocks assigned to it: the *Workflow Configurator* for design, the *Workflow Registry* for storing reusable templates, and the *Workflow Controller* for controlling execution. Through the *Annotation Editor*, medical experts review and refine the generated annotations. The processing steps themselves are represented as *Data Preparation Services* and *Annotation Services* with a multiplicity of 1..n. This notation captures the plugin architecture: both service types can be added, exchanged, and operated side by side without modifying the engine. In this way, the model directly addresses RC1 and RC2, as open-source LLMs appear as an annotation service without any change to the workflow definition. Externally, NERFlow connects to the remaining FIT4NER components: to KM-EP as the target environment (RC4), to the *NER Framework Independent Service*, through which the NER framework services 1..n are accessed, and to the registries for services, models, and model definitions. The representation thus makes visible what the tools examined in Section 2 do not provide: the creation of training data is not a separate step performed in a separate tool but is embedded in an environment that jointly manages documents, models, and workflows. The *NER Framework Independent Service* is not part of NERFlow itself but a building block introduced in the FIT4NER project that encapsulates framework-independent access to different NER tools [58]. NERFlow builds on it as an existing interface, through which the annotated data enter the training without binding NERFlow to a specific NER framework.

While Figure 2 shows the static structure, Figure 3 depicts the process flow. It is not a new development but the concrete realization of an already evaluated process model, namely, the *revised process model for AI-based knowledge extraction support for CPG development* [2]. That model describes the flow at a conceptual level. It was examined and revised in cognitive walkthrough sessions with expert reviewers until their findings had been resolved, and it was subsequently published. Figure 3 refines its abstract steps into a

concrete flow that can be executed with NERFlow, showing which steps documents pass through from selection to the reviewed training corpus, at which points decisions are made, and where medical experts intervene. This makes the decision points visible, for example, whether an annotation run is accepted, rejected, or repeated with a different workflow, along with the states of the data between the individual steps. The contribution of this model, therefore, lies not in the design of a new process but in the mapping of which concrete system step fulfills which abstract process step.

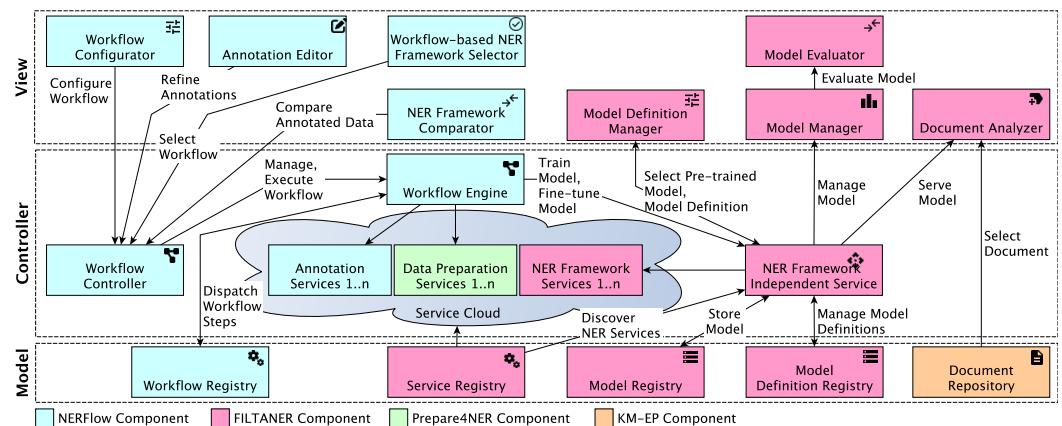


Figure 2. Placement of NERFlow within the integrated FIT4NER architecture.

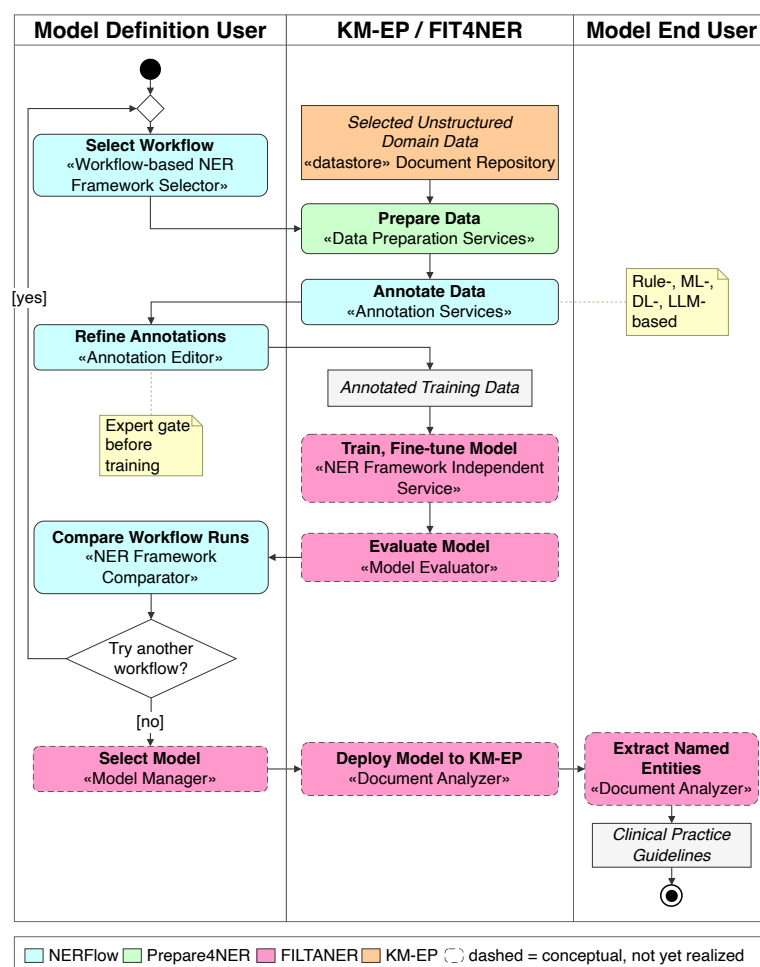


Figure 3. NERFlow-supported flow from document selection to the reviewed training corpus.

To illustrate subsystem interactions, Figure 4 shows the **Sequence Diagram for Loading Plugin Steps**. It demonstrates how Administrator Users register new plugin functionality, how metadata (type, inputs, and outputs) is persisted, and how the backend retrieves available plugin definitions for the workflow creator. This sequence highlights system extensibility and decoupling of plugins from workflow logic.

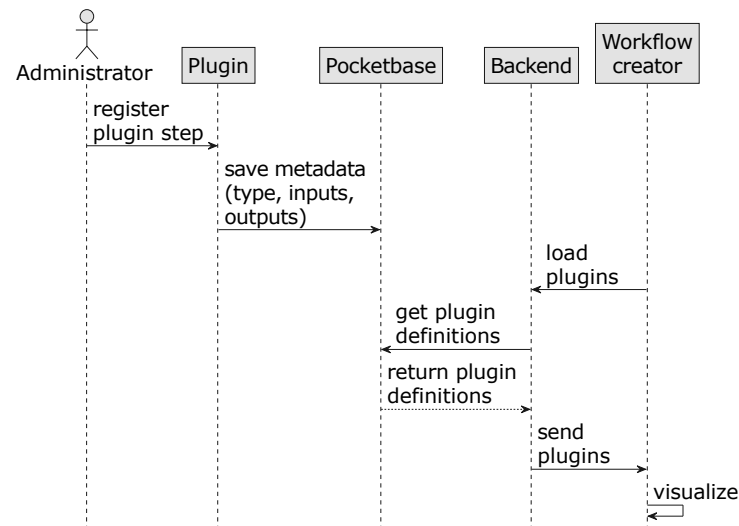


Figure 4. Sequence diagram for Loading Plugin Steps.

The **Microservice Subsystem** deploys and manages plugins as containerized services. Each microservice registers metadata (Docker image, version, maximum replicas, and K8S deployment templates) within the MicroServiceRegistry. Operations such as `startPod()`, `stopPod()`, and `getPodStatus()` are provided. Automated deployment via the K8S **Application Programming Interface (API)** ensures provider-independent, scalable execution (RC3). Integration into KM-EP (RC4) was evaluated through iframe embedding, full integration, and Web Components. Web Components offer the best trade-off: NERFlow's frontend is embedded as a reusable element, while backend communication occurs through an API gateway managed by Symphony, ensuring consistent authentication and styling.

Overall, NERFlow's models show how workflows are defined by administrators, executed and validated by domain experts, and supported by scalable microservice-based plugins. By integrating open-source LLMs, standardized workflows, K8S-based orchestration, and portal integration with KM-EP, NERFlow provides a robust and extensible platform for training medical NER models. This concludes the theory-building stage; the next section presents the implementation of the defined components.

4. Implementation

This section corresponds to the system development phase of the Nunamaker methodology and focuses on realizing the **Research Objective (RO)** through the implementation of NERFlow. Building on the models introduced in Section 3, the prototype implements a flexible, cloud-native system that supports the integration of open-source LLMs such as LLaMA, DeepSeek, Gemma, and Qwen. In line with the identified RCs, the implementation is organized around the three core subsystems of NERFlow: the Configuration Subsystem, the Execution Subsystem, and the Microservice Subsystem, followed by the integration into KM-EP.

The prototype combines modern web and NLP technologies chosen for scalability, interoperability, and maintainability. TypeScript 5.8 serves as the primary programming

language for the frontend, backend, and shared workflow components, providing static typing that enhances reliability and long-term refactorability. Python 3.11 is used selectively in NLP plugins that rely on existing frameworks such as spaCy 3.7 and Hugging Face Transformers 4.51. The overall architecture follows the **Model-View-Controller (MVC)** principle, with React 19-based components representing the View, Node.js 22, and Symphony 6.4 backends forming the Controller layer, and PocketBase 0.27 [59] acting as the Model layer and centralized data store for workflows, annotations, and metadata.

The **Configuration Subsystem** implements the conceptual configuration layer described in Section 3. It enables users to visually compose, configure, and manage annotation workflows. React was chosen for the frontend due to its component-based architecture and large ecosystem, which facilitate modular development of NERFlow. For workflow visualization, React Flow 12.5 is employed to represent nodes, inputs, outputs, and connections dynamically in the browser, mirroring the workflow models defined in the use context diagram. React-Bootstrap 2.10 ensures consistent styling based on established design patterns. For text annotation visualization and editing, an Angular 19-based web component was developed and embedded into the React application. This component wraps ngx-annotate-text 18 and provides a reusable, encapsulated annotation viewer. Angular was selected because it supports strongly typed, component-based development, making it suitable for the complex UI logic associated with interactive annotation. This hybrid approach demonstrates that NERFlow can integrate components from different frontend frameworks while maintaining a unified user experience. The backend of the Configuration Subsystem is implemented in TypeScript using Express 5.1, which exposes a **Representational State Transfer (REST)** API for workflow persistence, plugin registration, and database communication. PocketBase acts as a lightweight **Backend as a Service (BaaS)**, providing authentication, real-time updates, and storage in a single deployable unit. Listing 1 shows how a processing step is defined, using the Ollama plugin as an example. This structure lies at the core of the interchangeability that Section 2 identified as a gap. A plugin declares its inputs and outputs, the step types it is compatible with, and the configurable parameters, among them the language model to be used. The selectable models are stored in the `values` field so that switching models becomes a selection from this list and requires no intervention in the system. Figure 5 shows the Workflow Configurator, in which the available steps can be combined into a workflow (RC2).

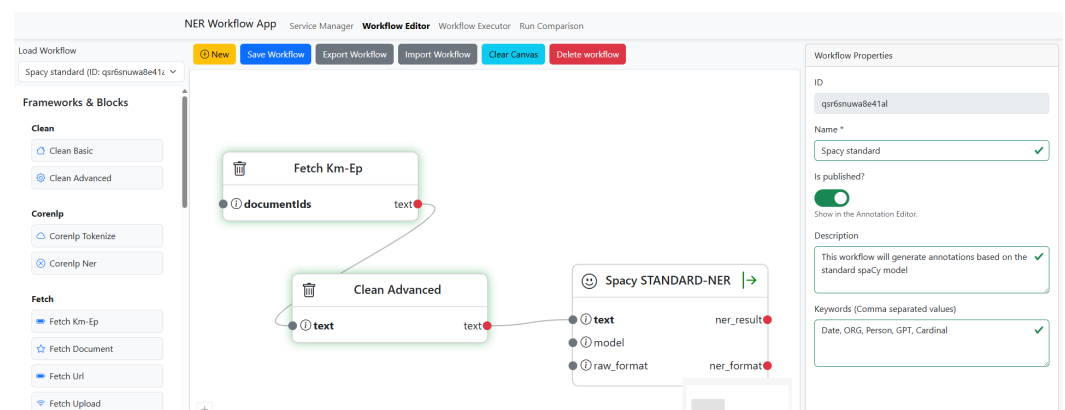


Figure 5. Workflow Configurator.

Listing 1. Step definition illustrated by the Ollama plugin.

```

1 const plugin: AnnotationPlugin = {
2   id: 'ollama',
3   steps: [{
4     step: 'ner',
5     category: StepCategory.Ner,
6     inputs: [
7       { name: 'text', type: 'text', required: true, compatible: ['text'] },
8       { name: 'model', type: 'text', required: true, compatible: ['text'],
9         default: "llama3.1:8b", isCustomizable: true,
10        values: ["llama3.1:8b", "deepseek-r1:14b", "gemma3:12b",
11               "qwen3:latest", "mistral:latest"] },
12       { name: 'labels', type: 'text', required: true, compatible: ['text'],
13         isCustomizable: true, default: "disease, symptom, treatment" },
14       { name: 'prompt', type: 'text', required: false, compatible: ['text'],
15         isCustomizable: true, default: examplePrompt }
16     ],
17     outputs: [{ name: 'entities', type: 'entities', required: false,
18               compatible: ['entities', "*"] }],
19     compatibleNodes: [StepCategory.Transform, StepCategory.Training],
20     process: async (payload: any) =>{
21       // Load the document content, process it, and~return the result
22       return new StepOutput(ner_result, "intermediate", documentId);
23     },
24   }],
25 };

```

The **Execution Subsystem** realizes the runtime behavior modeled in Section 3. It orchestrates workflow execution on selected documents. Model Definition Users select documents and workflows in the frontend; for LLM-based workflows, additional configuration options are available, such as selecting the underlying model, specifying custom labels, or providing a custom prompt. On the backend, workflow execution is delegated to a Workflow Engine built on the Inngest 3.35 [55] client, a serverless, event-driven orchestration platform. Plugins subscribe to updates in their dedicated PocketBase tables. When a data record is updated to the status processing, the plugin executes its `process()` function and writes the results back to the same table. The backend observes these changes and streams updates to the frontend using Socket.IO 4.8, providing responsive progress feedback to users. To use different NER frameworks within the same workflow, their outputs must be mapped into a shared intermediate format. Listing 2 shows this mapping using spaCy as an example. The transformation is expressed as a data structure rather than as program code, so connecting a further framework requires a format description and not an extension of the engine.

The **Microservice Subsystem** implements the container-based execution layer from the model. It manages isolated, containerized NLP plugins that encapsulate specific processing capabilities. Each plugin is packaged as a Docker container, providing clear boundaries, independent deployment, and cloud-agnostic execution. Plugins expose a `process()` function and subscribe to their PocketBase tables, ensuring that processing is triggered when input data become available and that outputs are written back reliably. Python-based plugins rely on spaCy and Hugging Face and are implemented as Flask 3.1 applications. Other NLP frameworks, such as Ollama or Gemini-based services, are integrated directly in TypeScript. CoreNLP 4.5.9, implemented in Java (OpenJDK 17), is accessed through `execSync()` calls from the TypeScript backend. This hybrid strategy reflects RC1 by enabling the integration of heterogeneous, open-source LLM and NLP frameworks without constraining the overall system to a single technology stack. The microservice manager,

implemented in TypeScript with React-Bootstrap, provides a graphical interface for starting, configuring, and stopping microservices. On the backend, the microservice manager communicates with the K8S API to create and manage deployments. K8S handles container orchestration, scaling, and fault tolerance, supporting local execution (Minikube) as well as deployment on AWS, GCP, or Microsoft Azure. This realizes the cloud-agnostic execution layer envisioned in RC3 and in the Microservice Subsystem model.

Listing 2. JSONata transform example for spaCy.

```

1 // Transform Example for SpaCy
2 const spacyTransform = {
3   framework: 'spacy',
4   toIntermediate: '
5     {
6       "text": $text,
7       "tokens": $tokens.[text],
8       "entities": $ents.{
9         "text": text,
10        "label": label,
11        "start": start_char,
12        "end": end_char
13      }
14    }
15  ',
16  toPlugin: '
17    {
18      "text": $text,
19      "tokens": $tokens.{"text": $},
20      "ents": $entities.{
21        "text": text,
22        "label": label,
23        "start_char": start,
24        "end_char": end
25      }
26    }
27  ',
28 };

```

Integration into KM-EP implements the portal-level embedding of NERFlow described in Section 3. NERFlow is exposed as a web component and embedded in Symphony/Twig templates, preserving modularity while enabling close interaction with the KM-EP ecosystem. Configuration data, such as user roles and content identifiers, are passed as HTML attributes at initialization time. KM-EP remains responsible for authentication and authorization, while NERFlow adapts its interface according to user roles (e.g., Model Definition User vs. Administrator User). The integrated frontend comprises three main components. The Annotation Wizard (Figure 6) guides users step-by-step through document selection, workflow configuration, and real-time monitoring of execution, including configuration of LLM-specific parameters. The Annotation Editor (Figure 7) provides visualization and manual refinement of annotations, while the Comparison Editor (Figure 8) supports side-by-side analysis of multiple annotation runs, merging of results, and export to BioC XML [60] for further processing. On the backend, additional Symphony controllers were introduced to connect NERFlow and KM-EP: the AnnotationController delivers the NERFlow frontend; the ExecutionController manages workflow execution, data exchange, and status updates; and the TaxonomyController synchronizes annotated entities with KM-EP taxonomies to improve search and retrieval. This integration implements RC4 by embedding workflow-based, LLM-assisted annotation directly into the existing knowledge management environment.

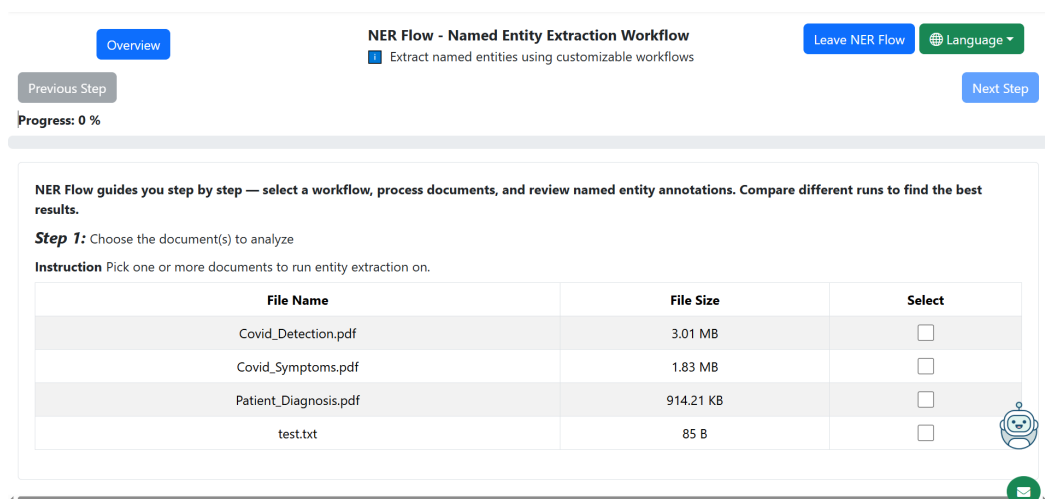


Figure 6. Annotation Wizard.

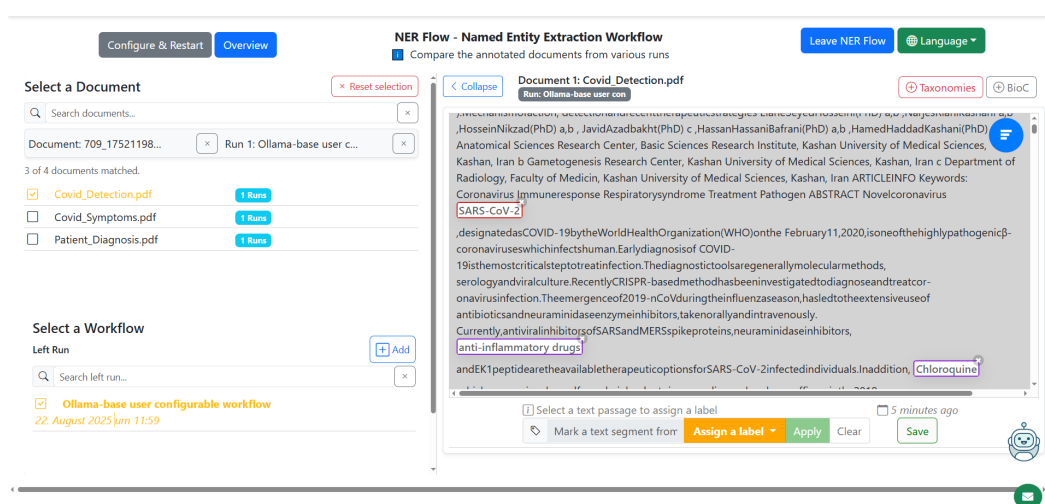


Figure 7. Annotation Editor.

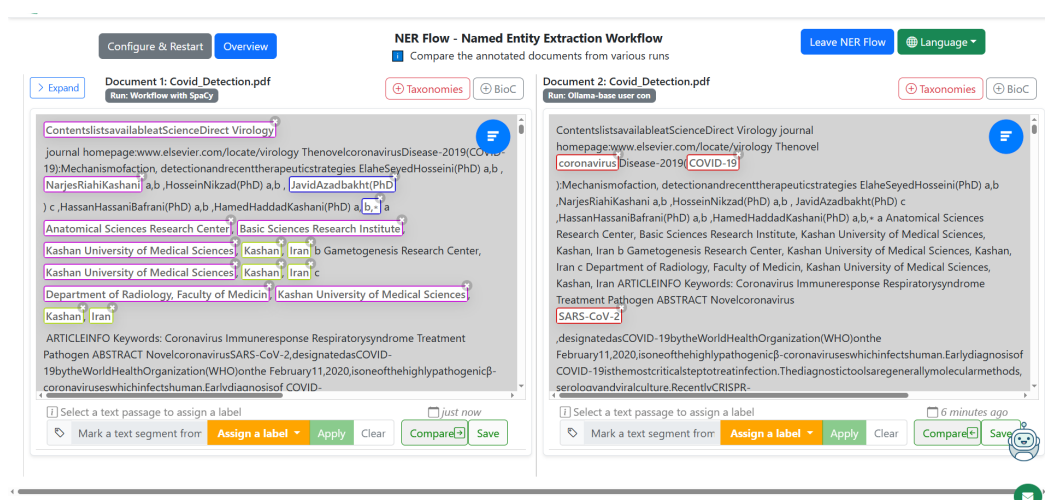


Figure 8. Comparison Editor.

The prototype thus operationalizes the conceptual design developed in Section 3 and addresses the RCs defined in Section 2. RC1 is addressed through the integration of open-source LLM-based plugins that support medical experts in generating training data. RC2 is fulfilled by the configurable workflow subsystem, which standardizes and structures

all preparatory steps for NER model training. RC3 is met by deploying all NLP services as K8S-managed containers, ensuring portability and cloud independence. Finally, RC4 is achieved through the seamless integration of NERFlow into KM-EP, enabling automated and LLM-assisted annotation directly within the platform.

With this, the implementation phase is complete, and the next section presents the evaluation of NERFlow.

5. Evaluation

The previous section detailed the implementation of a prototype designed to assist medical experts in creating training data for ML-based NER. This section corresponds to the experimentation phase of the Nunamaker methodology. It examines whether the implemented system meets the ROs and addresses the challenges identified in Section 2. The evaluation proceeds in three successive stages of increasing scope and differing focus: a formative usability assessment based on a **Cognitive Walkthrough (CW)**, a technical review following IEEE 1028, and a user-centered survey with 18 participants. These qualitative methods are complemented by a quantitative study of the integrated open-source language models on the CRAFT corpus.

5.1. Formative Evaluation: Cognitive Walkthrough

The CW [61] serves as a formative method for the early detection of usability problems. It adopts the perspective of inexperienced users and examines four questions for each work step: whether the goal is recognizable, whether the appropriate action can be found, whether that action can be associated with the goal, and whether the system's response is understood as progress. Two moderated sessions were held on 9 July and 18 July 2025. In addition to the author, four experts participated: a Ph.D. and habilitation candidate and three Ph.D. students. All possessed substantial knowledge of NLP and NER as well as in-depth familiarity with KM-EP. The tasks performed, and the corresponding findings, are summarized in Table 1.

Table 1. CW evaluation results.

Task	Description	Feedback and Findings
T1.1	Display workflow steps	The steps are already available as workflow nodes.
T1.2	Display transformation steps	Not considered in the current evaluation.
T1.3	Create a workflow	F1: Validation of input nodes is missing.
T1.4	Display documents	F2: File size information is missing.
T1.5	Select workflow	F3: Support for user-defined labels is requested.
T1.6	Compare runs	F4: Ensure consistent color mapping for labels.
T2.1	Display file size	OK.
T2.2	Present as a wizard	F5: Add descriptions and informational icons.
T2.3	Run-based user guidance	OK.
T2.4	Improved UI for comparison	F6: Limit comparison to two runs for clarity.
T2.5	Improved UI for merging	F7: Group labels; F8: Add batch merging functionality.
T2.6	Button to create taxonomies	OK.
T2.7	UI for BioC file	OK—considered a useful feature.

In the first evaluation session (T1.1–T1.6), the central recommendation was to align the user interface more closely with the needs of physicians and to provide clear guidance through the workflows (F1–F4). The technical foundations, in particular the annotation services as microservices and the configuration of the workflows, were already rated as functional. Based on these findings, an Annotation Wizard was introduced that guides users step-by-step through the entire annotation workflow and was positively received

during the second evaluation session (T2.1–T2.7). However, the user interface was still perceived as too complex. Participants recommended adding short textual descriptions to each wizard step and integrating information buttons (F5). In addition, the comparison of annotation runs should be simplified (F6–F8). Overall, the CW confirmed that NERFlow’s basic interaction concepts are suitable for Model Definition Users.

5.2. Technical Review

The usability assessment was complemented by a technical review. The technical review followed the IEEE 1028 standard [62] and examined architecture, code quality, documentation, security, maintainability, and testability. Installation was carried out on Windows 11 and macOS. A Ph.D. student with experience in IT operations, medical informatics, and NER worked through a checklist based on IEEE 1028 (Table 2). The review confirmed a clearly separated subsystem structure and highlighted the loose coupling via standardized interfaces as a strength. At the same time, it identified gaps that must be closed for use beyond a research prototype: the documentation needs improvement, there are no automated unit tests, the API endpoints are not authenticated, and the logging does not meet data protection requirements.

Table 2. Results of the technical review.

Task	Category	Description	Findings
TR1.1	Requirements	Are the requirements implemented in a comprehensible way?	Yes.
TR1.2	Requirements	Deviation between requirements and implemented solution?	Focus on flexibility and additional complex features; therefore challenging for the GUI.
TR2.1	Design	Is the system architecture clearly documented?	Yes.
TR2.2	Design	Are modules loosely coupled and clearly delineated?	Yes.
TR2.3	Design	Is the architecture extensible?	Yes—a strength of the system.
TR2.4	Design	Are data flows and interfaces consistent and traceable?	Yes.
TR3.1	Code Quality	Is the code well structured and modular?	Yes.
TR3.2	Code Quality	Are naming conventions and coding standards followed?	Largely yes.
TR3.3	Code Quality	Is the code free of duplication and redundant functions?	Largely yes.
TR3.4	Code Quality	Is error handling implemented systematically?	Yes.
TR4.1	Documentation	Are installation and user guides available?	Yes—including video material.
TR4.2	Documentation	Is the API documented and illustrated with examples?	Unclear: presence of API documentation is uncertain (may exist only in the thesis).
TR4.3	Documentation	Are configurations and workflows sufficiently described?	Yes.
TR4.4	Documentation	Are code comments present and helpful?	Very few; this area can be improved.
TR5.1	Security	Are access controls (e.g., user roles) in place?	Yes (provided by KM-EP). Question: can permissions also be managed within the NERFlow app?

Table 2. *Cont.*

Task	Category	Description	Findings
TR5.2	Security	Are API endpoints secured (e.g., authentication, tokens)?	No, but acceptable for a prototype.
TR5.3	Security	Are sensitive data protected (e.g., patient data, encryption)?	Transport encryption in place; not enforced at REST level but protected within KM-EP.
TR5.4	Security	Is logging compliant with data protection requirements?	No, but acceptable for a prototype.
TR6.1	Maintainability	Is the system easy to deploy?	The system is complex with many components but well-documented and deployable on Kubernetes.
TR6.2	Maintainability	Is versioning and repository management in place?	
TR6.3	Maintainability	Are external dependencies documented and up to date?	
TR6.4	Maintainability	Is the architecture prepared for future extensions?	
TR7.1	Testability	Can tests be executed automatically?	No unit tests found.
TR7.2	Testability	Is test coverage documented?	No unit tests found.
TR7.3	Testability	Are test data or benchmarks available (e.g., CRAFT Corpus)?	Yes.
TR7.4	Testability	Are results stored reproducibly?	Yes.

5.3. User-Centered Survey

To assess usability beyond the narrow circle of walkthrough participants, a structured survey was conducted following the process model of Kasunic [63], whose results were published in [64]. Participants first completed a guided task sequence, namely, selecting documents, running a workflow, reviewing and correcting annotations, and exporting results, and then filled out a trilingual questionnaire with 18 items, mostly on a seven-point Likert scale. The survey, carried out from 16 to 25 September 2025, involved 18 participants from Germany (10), Vietnam (7), and the United States (1), aged between 18 and 54 years. Twelve of them had no or only beginner-level experience with NER and six had advanced knowledge. Ten were students, and eight were working professionals. All items concerning usability, interface design, functional quality, and the NERFlow-specific features achieved mean values above 4.94. The highest ratings were given to login and document search in the portal (6.00) and to the help materials (5.56). The annotation assistant, introduced in response to the CW, received 5.50. In the free-text responses, participants repeatedly emphasized that the system noticeably reduces the manual annotation effort and that its guidance eases the entry for non-experts in particular. They named limits just as clearly: a missing change history, the lack of feedback when generating taxonomies, difficulties with very large annotation sets, and help materials that show click sequences without explaining their purpose. Two participants did not reach the task goals and gave consistently low ratings. The composition of the sample is decisive for the significance of the results. The participants had basic IT skills and were predominantly students and people from software development, complemented by isolated cases from research, marketing, and education. The survey, therefore, demonstrates that the system can be operated by people without prior NER knowledge.

5.4. Quantitative Evaluation of the Language Models

The basis is the CRAFT corpus [65] in version 5 (v5.0.2), comprising 97 biomedical full-text articles. The mapping to the MONDO ontology [66] was used, with 3650 annotated entities across 326 concepts. The target classes are the concept labels themselves, such as *Huntington's disease* or *movement disorder*. For each document, the model receives in its prompt the list of MONDO concepts occurring in that document, which across all documents is exactly the annotated gold set (on average, 6 of the 326). Unlike a trained model that has to infer the concepts from the text, the model is told in advance which concepts appear in the document and only has to find their textual occurrences. The values, therefore, measure *localization* given a predefined set of concepts and are not comparable to open NER. This design follows the system architecture, in which the target terms are a workflow parameter specified by the domain expert, and a complete ontology cannot be entered manually for each document. Because all models receive the same specification, reproducibly derived from the corpus, their comparison remains fair, and the downstream expert review catches remaining errors.

Seven language models were evaluated (Table 3): five open-source models operated locally via Ollama [67] and two commercially hosted models accessed through a vendor-independent gateway interface. All ran through the same NERFlow processing chain, that is, splitting into segments of 5000 characters, an enforced output schema, and a uniform system prompt so that performance differences can be attributed to the models themselves. Each model is based on a single run with the provider's default settings, without a set temperature or fixed random seed. The two hosted models stand as examples of cloud operation. The goal is not to determine the best commercial model but to show that the same interchangeable architecture works without modification both locally and in the cloud. The prompt, output schema, segment size, corpus version, and matching procedure are fully fixed, and the evaluation is reproducible via a documented, checksum-secured pipeline. The prompt and model configuration are documented in Appendix A.

Table 3. Evaluated language models: local open-source and commercially hosted.

Model	Identifier	Deployment	Parameters
Llama 3.1	llama3.1:8b	local (Ollama)	8B
Mistral	mistral:latest	local (Ollama)	7.2B
Qwen 3	qwen3:latest	local (Ollama)	8.2B
Gemma 3	gemma3:12b	local (Ollama)	12B
DeepSeek-R1	deepseek-r1:14b	local (Ollama)	14B
Claude Haiku 4.5	hosted	Gateway	proprietary
Claude Sonnet 4.6	hosted	Gateway	proprietary

Evaluation uses **Precision (P)**, **Recall (R)**, and **F-Score (F1)** [68]. The *micro* measure is reported, aggregating over all instances [69]. A class-wise *macro* average is unsuitable here, because a considerable portion of the 326 concepts in the corpus is represented by only a single entity, so a class average would predominantly reflect the variance of rare individual cases. A match requires agreement in the surface form, that is, the wording of the occurrence in the text, and in the concept label. Because the models rarely reproduce the wording character for character, it is checked in a similarity-based manner, using the Jaro-Winkler similarity in both dimensions with an empirically determined threshold of $\tau = 0.60$ [70]. Lower values raise all results and can change the ranking, so similarity-based values can only be interpreted together with τ . Results are reported at two levels. At the *mention level*, a match counts as soon as the concept and label agree, regardless of the text position. At the *span level*, the position must additionally agree, meaning the exact start

value or an overlap with the gold span. This distinction is central because the annotations are transferred to an editor for correction, and their positioning is therefore of practical relevance. Two further specifications apply to the matching. First, the assignment is one-to-one, so each reference annotation serves as a match at most once. Second, entries that cannot be evaluated on either side are excluded beforehand. Among the predictions, these are entries with an out-of-domain label or an unusable position. Among the references, there are 69 entries with an empty text field, which cannot be assigned due to a lack of wording. Almost all of them stem from multi-part gold annotations with a gap, such as “agenesis of ... corpus callosum”, which the corpus preparation merges into a single span, emptying their text in the process.

Table 4 shows the micro values at the mention level. The highest result is achieved by the hosted Claude Sonnet 4.6 with an F1 of 0.637, followed by Claude Haiku 4.5 with 0.537. Among the locally operated open-source models, Gemma 3 12b (0.361) leads ahead of Qwen 3 (0.298) and Llama 3.1 (0.295). Mistral (0.252) and DeepSeek-R1 14b (0.096) follow. DeepSeek-R1 stands out for its pronounced reticence: a comparatively high precision (0.411) is offset by a recall of only 0.054. The reasoning model outputs very few entities.

Table 4. Micro results at mention level, similarity-based matching ($\tau = 0.60$).

Model	P	R	F1
<i>local (open-source)</i>			
Llama 3.1	0.344	0.258	0.295
Mistral	0.254	0.250	0.252
Qwen 3	0.249	0.373	0.298
Gemma 3 12b	0.300	0.455	0.361
DeepSeek-R1 14b	0.411	0.054	0.096
<i>hosted</i>			
Claude Haiku 4.5	0.637	0.464	0.537
Claude Sonnet 4.6	0.730	0.564	0.637

Note: Italics indicate model deployment category (local open-source vs. hosted); bold indicates the highest F1 score.

The hosted models have a limitation that the local ones do not. With Claude Sonnet, the provider’s content filter rejected three of 868 segments of a single document (PubMed 14609438, an article on the enzyme tripeptidyl-peptidase I) with empty text and the justification *content filter*, reproducibly and without the option to disable it. For this harmless subject, the number is small, but for sensitive medical topics, such a filter may trigger, depending on the topic, and distort the measurement without any relation to model capability. For a reproducible, topic-independent run, local open-source models are therefore the more reliable basis, which supports the vendor-independent architecture.

At the span level, an implementation error in the position reconstruction of the processing chain came to light, which the evaluation uncovered. The chain splits long documents into segments of 5000 characters and reconstructs the position of each entity through a text search *within its segment*, without adding the offset of the segment start. As a result, all positions from the second segment onward are systematically wrong, too far forward by a multiple of 5000 characters. The finding is documented in the source code and in the stored outputs: almost all start values lie below the first segment boundary, where they would have to be broadly distributed given an average document length of 42,180 characters. The error was corrected by adding the segment offset to the position. The correction requires no new model call, since it operates on the already stored outputs. Its effect on the span F1 is considerable (Table 5): across all models it rises to roughly three times its value.

Claude Sonnet rises from 0.171 to 0.500, Claude Haiku from 0.125 to 0.376, Gemma 3 12b from 0.055 to 0.177, and Llama 3.1 from 0.026 to 0.080.

The gain has a model-dependent upper bound. The correction can only reposition occurrences whose text location was found in the segment at all. If a model returns labels from the predefined label list without the term appearing in the text, the position remains an invention onto which the offset is merely added further. This findability rate is the upper bound of the gain. Across the corpus, it ranges from 93.5 percent for Claude Sonnet and 92.7 percent for Claude Haiku down to 65 to 68 percent for Llama 3.1, DeepSeek-R1, and Mistral. The more disciplined a model localizes, the fewer predictions remain uncorrectable, and the greater the gain. Claude Sonnet gains +0.329, Mistral only +0.040. Even after the correction, the span F1 remains below the mention value (Claude Sonnet 0.500 versus 0.637). The remaining gap stems from two causes independent of the offset: the predefined label list, which allows concepts to be reported that do not appear in the text, and the merging of discontinuous gold annotations into a single span.

Table 5. Effect of the segment-offset correction on the span-level matching (similarity-based, $\tau = 0.60$, with position check).

Model	Before Correction	After Correction	Δ
Claude Sonnet 4.6	0.171	0.500	+0.329
Claude Haiku 4.5	0.125	0.376	+0.251
Gemma 3 12b	0.055	0.177	+0.122
Qwen 3	0.032	0.105	+0.073
Llama 3.1	0.026	0.080	+0.054
Mistral	0.022	0.062	+0.040
DeepSeek-R1 14b	0.011	0.034	+0.023

Note: Bold indicates the highest value after correction.

This finding shifts the interpretation relative to the literature. It has been reported that the position values output by language models are often unusable [19]. Here, however, the drastic span drop before the correction (Table 5) was mostly due to the missing offset addition in the processing chain and not to an inability of the models. After the correction, the positioning is as good as the model localizes the term in the segment. The path suggested there, namely, providing the model with the token positions already in the input, remains a sensible next step that the present implementation does not pursue.

For the intended use of the system, what matters is how the remaining mislocalizations affect the annotation editor. With the segment-offset correction, the marking of a recognized entity sits on an actual occurrence of its term within the same segment, that is, in the context that the domain expert reads anyway. The remaining cases are predictions whose term does not appear in the segment at all. They are false positives that the expert review discards, not positions that need fine-tuning. Their proportion is the complement of the findability rate and ranges from about 7 percent for the Claude models to roughly one-third for the weakest local models. Because the annotations are reviewed before inclusion in a training corpus in any case, such a mislocalization weighs less heavily than a missed or mistyped entity that would have to be found manually. The results thus support operation as pre-annotation with expert correction, not as autonomous annotation.

A rough comparison with trained recognition methods is possible on the same corpus. In a preceding work of the same research project, framework-supported NER models were trained and evaluated on CRAFT v5 [58]. The spaCy framework, which performed most consistently there, achieves F1 values around 90 percent across the ChEBI, CL, PR, and UBERON ontologies (90.27, 90.22, 89.87, and 90.11 percent). These values, however, are based on ontologies other than the MONDO mapping used here and on a different

evaluation strategy, so they are only comparable to a limited extent. The gap to the values reported here is considerable and should not be downplayed. It points, however, to different roles: the trained models presuppose an annotated corpus, whose creation the system presented here is meant to support. LLM-supported pre-annotation does not compete with them but precedes the training phase.

That the lower quality per annotation does not negate the benefit is grounded in the literature (Section 2). Ding et al. [43] show that a model trained on language-model-annotated data often reaches or exceeds the human baseline at a *fixed budget*, because considerably more data can be annotated for the same effort. The advantage arises from quantity, not from accuracy per individual case. For local open-source operation, the limiting factor shifts from cost to computation time and model quality, but the basic argument remains: pre-annotation with subsequent review reduces the manual effort of corpus creation, as demonstrated for dictionary-based methods [42]. Whether this holds unchanged for language models is an open question, but the error structure measured here supports the use, as mentions are mostly recognized and the positions are correctable.

5.5. Limitations of the Study

The significance of the quantitative study is limited in three respects. The *scope of the task* limits the results. Because the predefined concept list corresponds to the annotated gold set, the study measures the localization of predefined concepts, not open NER. The values are therefore an optimistic upper bound, restricted to the MONDO mapping and not readily transferable to other ontologies. A fair comparison with trained models under an open, complete set of concepts remains reserved for future work. In addition, two effects independent of the corrected position error weigh on span accuracy: the predefined label list and the merging of discontinuous gold annotations. The *control of the measurement* is incomplete. Each model is based on a single run with the provider's defaults. Without a fixed temperature or fixed random seed, the stochasticity is not controlled, so the absolute values are subject to a certain variance. Predictions and reference annotations are filtered before matching, and for the hosted models, a provider-side content filter is added that is neither controllable nor versioned. The *reach of the evidence* is limited. No medical professionals participated in the user-centered survey. It demonstrates operability by people without prior NER knowledge, not suitability for the clinical professional context. Whether the generated pre-annotations actually improve a downstream training is not examined by this work. The actual purpose of the system, therefore, remains indirectly supported. In addition, a dedicated comparison baseline trained on an identical ontology is lacking. The classification relative to trained models relies on differing ontologies of the same document set. Finally, the interchangeability across annotation paradigms that the architecture provides is established by construction through the plugin design (Section 4). The quantitative study compares open-source and hosted LLMs with one another, whereas a systematic empirical comparison of rule-based, model-based, and LLM-based annotation on the same corpus remains future work.

6. Conclusions

In this work, a system was designed, implemented, and evaluated following the structured Nunamaker methodology to support medical experts in creating training data for ML-based NER models. The resulting prototype, NERFlow, enables users to run LLM-based annotation workflows, compare and correct results, and integrate the annotated data into the KM-EP knowledge management environment. As set out in Section 1, the contribution lies not in a single annotation method but in the architecture that makes such methods interchangeable. Through an abstraction layer, rule-based, model-based, and LLM-based

annotation can be deployed and compared within the same workflow-based environment, and within LLM-based annotation, different open-source models can be used as well, without modifying the system.

The *Introduction and Motivation* Section positioned the work within evidence-based medicine and clinical guideline development, formulated two RQs, and motivated the use of open-source LLMs in a cloud-agnostic setting. The *Related Work* Section identified four RCs: (RC1) support for training data generation through LLMs, (RC2) standardization of preparatory steps via workflows, (RC3) the use of K8S as a cloud-agnostic execution layer, and (RC4) integration into KM-EP. The *Modeling* Section translated these challenges into conceptual models for configuration, execution, and microservice subsystems and defined their integration with KM-EP. The *Implementation* Section described how these models were realized using TypeScript, Python-based plugins, event-driven workflow orchestration, and K8S-managed microservices. The *Evaluation* Section applied qualitative and quantitative methods to assess usability, technical soundness, and LLM performance. CWs and a technical review provided expert feedback on interaction design, architecture, and maintainability, while quantitative experiments on the CRAFT corpus measured the quality of automated annotations.

The evaluation supports the overall design of this work both qualitatively and quantitatively. The CW confirmed the fundamental interaction concepts, and a survey [64] with 18 participants confirmed usability even without prior knowledge of NER. The technical review following IEEE 1028 attested to a loosely coupled architecture with gaps that remain open for productive use, namely, unit testing, authentication, and logging. Quantitatively, the annotation quality remains limited, yet the central result is the demonstrated interchangeability: seven LLMs, five local and two hosted, passed through the same environment without modification and could be compared directly. The best mention-level F1 is achieved by the hosted Claude Sonnet 4.6 (0.637 versus 0.361 for Gemma 3 12b), but a content filter that cannot be disabled discarded parts of a harmless document so that, for reproducible operation, the locally deployable models are the more reliable basis. In this way, the architecture carries the exchange of methods, preserves their value as models advance, and allows providers to be chosen per requirement, whether reproducibility or peak performance.

Taken together, the results demonstrate that NERFlow provides an integrated abstraction layer and multiple methods for automated annotation generation using different LLMs, suitable for training with the selected NER frameworks, thereby answering RQ1. Furthermore, the system enables the management and configuration of heterogeneous annotation services with different parameters and execution environments, thereby addressing RQ2. At the same time, the prototype operationalizes the four RCs: open-source LLMs are integrated as annotation services (RC1), preparatory steps are modeled and executed as standardized workflows (RC2), workflows and plugins run on K8S as a cloud-agnostic layer (RC3), and the entire functionality is embedded into KM-EP (RC4).

The outlook follows directly from the design of this work. Most pressing is an investigation of the *end-to-end* process. The present evaluation examines usability, technical aspects, user judgment, and model quality but not the complete execution of the process model [2] underlying Section 3, from document selection to the reviewed corpus. A study with medical professionals covering this entire run, therefore, remains open, including the actual time savings compared to purely manual annotation. It would, at the same time, answer the question of the correction effort caused by the low positional accuracy. A second starting point concerns precisely these positions. LLMs report character positions unreliably because they would ultimately have to count them. Instead of reconstructing the positions afterward through text search, the input could therefore be annotated in advance

with a position marker for each token. The model would then no longer need to compute the position itself but only name the matching marker [19].

Beyond these focus areas, the system can be extended to additional NER frameworks, further open-source LLMs, and additional data modalities such as audio and video. Usability and efficiency could be improved by intelligent assistance features, richer visualizations, history and provenance tracking, and automatic error detection or suggestion mechanisms. From an operational perspective, tighter integration with cloud-native services would enable automated scaling, stronger security guarantees, and stable long-term operation in clinical or research infrastructures.

Overall, this work answers the RQs posed and addresses the RCs identified. NER-Flow appears less as a finished annotation tool than as a flexible, extensible, and vendor-independent platform whose actual value lies in the interchangeability of the annotation methods, and which thereby supports medical professionals in creating training data for NER models.

Author Contributions: Conceptualization, F.F., P.T., B.T. and M.H.; methodology, F.F., P.T., B.T. and M.H.; software, F.F., P.T. and B.T.; validation, F.F., P.T. and M.H.; formal analysis, F.F., P.T. and M.H.; investigation, M.H.; resources, F.F., P.T. and M.H.; data curation, F.F., P.T. and M.H.; writing—original draft preparation, F.F. and B.T.; writing—review and editing, P.T.; visualization, F.F., P.T. and M.H.; supervision, M.H.; project administration, F.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Acknowledgments: During the preparation of this manuscript, the authors used Anthropic Claude (Opus 4.8, Anthropic PBC, accessed via Claude Code) for building a reproducible evaluation harness around the original, unmodified annotation and scoring code, for data analysis and interpretation, for preparing the source file of one figure, and for drafting and language editing. The evaluation runs were executed by the authors, all reported figures derive from the original code, and all references were verified against their primary sources. The authors have reviewed and edited all AI-assisted output and take full responsibility for the content of this publication. Note that Claude models also appear as evaluation subjects in the evaluation (Claude Sonnet 4.6 and Claude Haiku 4.5), which is distinct from this authoring assistance.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. System Prompt and Model Configuration

This appendix documents how the LLMs were driven in the condition with an enforced output schema (Section 5). Listing A1 reproduces the system prompt verbatim. The placeholder `${labels}` is replaced per document by the list of admissible labels derived from the MONDO ontology, precisely the specification that turns the task into a localization of predefined concepts. The document text of the respective segment is appended as the user message.

The output is enforced through a schema (Structured Outputs): the schema shown in Listing A2 is translated to JSON Schema and passed to the model as a format constraint so that the response is a list of objects with `text`, `label`, `start`, and `end`. Beyond this, the model call sets no parameters, in particular, no temperature and no random seed. The respective provider's defaults apply. Before submission, the documents are split into segments of 5000 characters.

Listing A1. System prompt of the schema-bound annotation (verbatim; `{labels}` is replaced per document by the MONDO labels).

```

1 Your task is to extract named entities from the provided document text.
2 The document content is included below.
3 The annotation labels for named entities are restricted to the following
4 list: {labels}.
5
6 Only include entities that clearly match the specified labels or standard
7 named entity types if no labels are provided.
8
9 If a text or part of it is already found as an named entity, do not
10 annotate it again.
11 If no named entities are found, return an empty annotation list and note
12 this in the response.
13
14 Provide the output in a structured format (e.g., JSON or a clear list)
15 for readability.
16
17 The following json is only example. Don't add person or location to the
18 labels.
19
20 Example Output Format:
21 json
22 [
23     { "label": "Person", "text": "John Doe", "start": 0, "end": 8 },
24     { "label": "Location", "text": "New York", "start": 15, "end": 23 }
25 ]

```

Listing A2. Enforced output schema. Defined as a Zod schema, translated to JSON Schema and set as the format constraint of the model call.

```

1 AnnotationSchema = array of objects:
2     text : string
3     label : string
4     start : number
5     end : number
6
7 Model call: chat({ model, messages, format: JSONSchema(AnnotationSchema) })
8             no temperature, no seed (provider defaults)
9 Segment size: 5000 characters

```

References

1. Lu, Q.; Li, R.; Wen, A.; Wang, J.; Wang, L.; Liu, H. Large Language Models Struggle in Token-Level Clinical Named Entity Recognition. *AMIA Annu. Symp. Proc.* **2024**, *2024*, 748–757.
2. Freund, F.; Tamla, P.; Hemmje, M. Towards improving clinical practice guidelines through named entity recognition: Model development and evaluation. In Proceedings of the 2023 31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS), Letterkenny, Ireland, 7–8 December 2023; pp. 1–8. <https://doi.org/10.1109/AICS60730.2023.10470480>.
3. Graham, R.; Mancher, M.; Wolman, D.M.; Greenfield, S.; Steinberg, E. (Eds.) *Clinical Practice Guidelines We Can Trust*; National Academies Press: Washington, DC, USA, 2011. <https://doi.org/10.17226/13058>.
4. Kuhlen, R. *Informationsethik: Umgang mit Wissen und Information in Elektronischen Räumen*; Number 2454 in UTB; UVK Verlagsgesellschaft: Konstanz, Germany, 2004.
5. Wen, C.; Chen, T.; Jia, X.; Zhu, J. Medical Named Entity Recognition from Un-labelled Medical Records Based on Pre-trained Language Models and Domain Dictionary. *Data Intell.* **2021**, *3*, 402–417. https://doi.org/10.1162/dint_a_00105.
6. Li, J.; Sun, A.; Han, J.; Li, C. A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 50–70. <https://doi.org/10.1109/TKDE.2020.2981314>.
7. Zhui, L.; Fenghe, L.; Xuehu, W.; Qining, F.; Wei, R. Ethical Considerations and Fundamental Principles of Large Language Models in Medical Education: Viewpoint. *J. Med. Internet Res.* **2024**, *26*, e60083. <https://doi.org/10.2196/60083>.

8. DFG Priority Programme RATIO (SPP 1999). RATIO: Rationalizing Recommendations (RecomRatio). 2017. Available online: <https://spp-ratio.de/projects/recomratio/> accessed on 31 July 2026.
9. FTK. FTK e.V. Research Institute for Telecommunications and Cooperation. 2023. Available online: <https://www.ftk.de/en> accessed on 31 July 2026.
10. Lamberth-Cocca, S.; Dimanova, V.; Bruchhaus, S.; Nawroth, C.; Mc Kevitt, P.; Hemmje, M. Towards Robust Named Entity Recognition to Support the Extraction of Emerging Technological Knowledge from Biomedical Literature. In *Proceedings of the CERC 2023*; Hochschule Darmstadt: Darmstadt, Germany, 2023. https://doi.org/10.48444/h_docs-pub-518.
11. FTK. *Artificial Intelligence for Hospitals, Healthcare & Humanity (AI4H3)*; R&D White Paper; FTK e.V. Research Institute for Telecommunications and Cooperation: Dortmund, Germany, 2020.
12. Vu, B.; Wu, Y.; Afli, H.; Mc Kevitt, P.; Walsh, P.; Engel, F.; Fuchs, M.; Hemmje, M. A Metagenomic Content and Knowledge Management Ecosystem Platform. In *Proceedings of the 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, San Diego, CA, USA, 18–21 November 2019; pp. 1–8.
13. Tamla, P.; Hartmann, B.; Nguyen, N.; Kramer, C.; Freund, F.; Hemmje, M. CIE: A Cloud-Based Information Extraction System for Named Entity Recognition in AWS, Azure, and Medical Domain. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*; Coenen, F., Fred, A., Aveiro, D., Dietz, J., Bernardino, J., Masciari, E., Filipe, J., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 127–148.
14. Freund, F.; Tamla, P.; Reis, T.; Hemmje, M.; Kevitt, P.M. FIT4NER—Towards a Framework-Independent Toolkit for Named Entity Recognition. In *Proceedings of the CERC 2023*; Hochschule Darmstadt: Darmstadt, Germany, 2023. https://doi.org/10.48444/h_docs-pub-518.
15. Tamla, P.; Freund, F.; Hemmje, M. SNERC: Enhancing Knowledge Management with Named Entity Recognition and Document Classification for Apply Gaming. *Artif. Intell. Appl.* **2025**, *3*, 392–407. <https://doi.org/10.47852/bonviewAIA52023841>.
16. Ng, M.Y.; Helzer, J.; Pfeffer, M.A.; Seto, T.; Hernandez-Boussard, T. Development of Secure Infrastructure for Advancing Generative Artificial Intelligence Research in Healthcare at an Academic Medical Center. *J. Am. Med. Inform. Assoc.* **2025**, *32*, 586–588. <https://doi.org/10.1093/jamia/ocaf005>.
17. Schneider, J. Explainable Generative AI (GenXAI): A Survey, Conceptualization, and Research Agenda. *Artif. Intell. Rev.* **2024**, *57*, 289. <https://doi.org/10.1007/s10462-024-10916-x>.
18. Wang, S.; Sun, X.; Li, X.; Ouyang, R.; Wu, F.; Zhang, T.; Li, J.; Wang, G. GPT-NER: Named Entity Recognition via Large Language Models. *arXiv* **2023**, arXiv:2304.10428.
19. Zeghidi, H.; Moncla, L. Evaluating Named Entity Recognition Using Few-Shot Prompting with Large Language Models. In *Proceedings of the A Conversation Between AI and the Humanities*, Lyon, France, 14–15 November 2024.
20. Apache Software Foundation. Apache Airflow Documentation. 2025. Available online: <https://airflow.apache.org/> (accessed on 31 July 2026).
21. Temporal Technologies. Temporal Platform Documentation. 2025. Available online: <https://docs.temporal.io/> (accessed on 31 July 2026).
22. Jawarneh, I.M.A.; Bellavista, P.; Bosi, F.; Foschini, L.; Martuscelli, G.; Montanari, R.; Palopoli, A. Container Orchestration Engines: A Thorough Functional and Performance Comparison. In *Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 20–24 May 2019; pp. 1–6. <https://doi.org/10.1109/ICC.2019.8762053>.
23. Tran, B. Entwicklung einer Microservice-Basierten Anwendung zur Automatisierten Annotation von Medizinischen Forschungsergebnissen für Named Entity Recognition. Master's Thesis, FernUniversität in Hagen, Hagen, Germany, 2025.
24. Nunamaker, J.F., Jr.; Chen, M.; Purdin, T.D.M. Systems Development in Information Systems Research. *J. Manag. Inf. Syst.* **1990**, *7*, 89–106. <https://doi.org/10.1080/07421222.1990.11517898>.
25. Jehangir, B.; Radhakrishnan, S.; Agarwal, R. A survey on Named Entity Recognition—Datasets, tools, and methodologies. *Nat. Lang. Process. J.* **2023**, *3*, 100017. <https://doi.org/10.1016/j.nlp.2023.100017>.
26. Pagad, N.S.; Pradeep, N. Clinical named entity recognition methods: An overview. In *Proceedings of the International Conference on Innovative Computing and Communications*, New Delhi, India, 19–20 February 2022; pp. 151–165.
27. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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*, Minneapolis, MI, USA, 2–8 June 2019; pp. 4171–4186. <https://doi.org/10.18653/v1/N19-1423>.
28. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI* **2019**, *1*, 9.
29. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
30. OpenAI. ChatGPT. 2025. Available online: <https://chatgpt.com/> (accessed on 31 July 2026).

31. Fink, A.; Rau, A.; Kotter, E.; Bamberg, F.; Russe, M.F. Optimierte Interaktion Mit Large Language Models. *Die Radiol.* **2025**, *65*, 235–242. <https://doi.org/10.1007/s00117-025-01416-2>.
32. Stenetorp, P.; Pyysalo, S.; Topić, G.; Ohta, T.; Ananiadou, S.; Tsujii, J. BRAT: A Web-Based Tool for NLP-Assisted Text Annotation. In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, 23–27 April 2012; pp. 102–107.
33. Klie, J.C.; Bugert, M.; Boullosa, B.; Eckart de Castilho, R.; Gurevych, I. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations, Santa Fe, NM, USA, 20–26 August 2018; pp. 5–9.
34. Giachelle, F.; Irrera, O.; Silvello, G. MedTAG: A Portable and Customizable Annotation Tool for Biomedical Documents. *BMC Med. Inform. Decis. Mak.* **2021**, *21*, 352. <https://doi.org/10.1186/s12911-021-01706-4>.
35. Explosion AI. Prodigy: An Annotation Tool for AI, Machine Learning and NLP. 2024. Available online: <https://prodi.gy/> (accessed on 31 July 2026).
36. Ogren, P.V. Knowtator: A Protégé Plug-in for Annotated Corpus Construction. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Demonstrations, New York, NY, USA, 4–9 June 2006; pp. 273–275. <https://doi.org/10.3115/1225785.1225791>.
37. Bada, M.; Vasilevsky, N.; Baumgartner, W.A.; Haendel, M.; Hunter, L.E. Gold-Standard Ontology-Based Anatomical Annotation in the CRAFT Corpus. *Database* **2017**, *2017*, bax087. <https://doi.org/10.1093/database/bax087>.
38. Bodenreider, O. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Nucleic Acids Res.* **2004**, *32*, D267–D270. <https://doi.org/10.1093/nar/gkh061>.
39. Savova, G.K.; Masanz, J.J.; Ogren, P.V.; Zheng, J.; Sohn, S.; Kipper-Schuler, K.C.; Chute, C.G. Mayo Clinical Text Analysis and Knowledge Extraction System (cTAKES): Architecture, Component Evaluation and Applications. *J. Am. Med. Inform. Assoc.* **2010**, *17*, 507–513. <https://doi.org/10.1136/jamia.2009.001560>.
40. Schäfer, H.; Idrissi-Yaghir, A.; Arzideh, K.; Damm, H.; Pakull, T.M.; Schmidt, C.S.; Bahn, M.; Lodde, G.; Livingstone, E.; Schadendorf, D.; et al. BioKGrapher: Initial Evaluation of Automated Knowledge Graph Construction from Biomedical Literature. *Comput. Struct. Biotechnol. J.* **2024**, *24*, 639–660. <https://doi.org/10.1016/j.csbj.2024.10.017>.
41. Comeau, D.C.; Islamaj Doğan, R.; Ciccicarese, P.; Cohen, K.B.; Krallinger, M.; Leitner, F.; Lu, Z.; Peng, Y.; Rinaldi, F.; Torii, M.; et al. BioC: A Minimalist Approach to Interoperability for Biomedical Text Processing. *Database* **2013**, *2013*, bat064. <https://doi.org/10.1093/database/bat064>.
42. Lingren, T.; Deleger, L.; Molnar, K.; Zhai, H.; Meinzen-Derr, J.; Kaiser, M.; Stoutenborough, L.; Li, Q.; Solti, I. Evaluating the Impact of Pre-Annotation on Annotation Speed and Potential Bias: Natural Language Processing Gold Standard Development for Clinical Named Entity Recognition in Clinical Trial Announcements. *J. Am. Med. Inform. Assoc.* **2014**, *21*, 406–413. <https://doi.org/10.1136/amiajnl-2013-001837>.
43. Ding, B.; Qin, C.; Liu, L.; Chia, Y.K.; Li, B.; Joty, S.; Bing, L. Is GPT-3 a Good Data Annotator? In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023. <https://doi.org/10.18653/v1/2023.acl-long.626>.
44. Hu, Y.; Chen, Q.; Du, J.; Peng, X.; Kelothe, V.K.; Zuo, X.; Zhou, Y.; Li, Z.; Jiang, X.; Lu, Z.; et al. Improving Large Language Models for Clinical Named Entity Recognition via Prompt Engineering. *J. Am. Med. Inform. Assoc.* **2024**, *31*, 1812–1820. <https://doi.org/10.1093/jamia/ocad259>.
45. Minaee, S.; Mikolov, T.; Nikzad, N.; Chenaghlu, M.; Socher, R.; Amatriain, X.; Gao, J. Large Language Models: A Survey. *arXiv* **2025**, arXiv:2402.06196. <https://doi.org/10.48550/arXiv.2402.06196>.
46. Ateia, S.; Kruschwitz, U. Can Open-Source LLMs Compete with Commercial Models? Exploring the Few-Shot Performance of Current GPT Models in Biomedical Tasks. In *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024)*; Faggioli, G., Ferro, N., Galuščáková, P., García Seco de Herrera, A., Eds.; CEUR-WS.org: Aachen, Germany, 2024; Volume 3740 CEUR Workshop Proceedings, pp. 78–98.
47. Freund, F.; Tamla, P.; Hemmje, M. Survey: Understand the Challenges of Machine Learning Experts Using Named Entity Recognition Tools. In Proceedings of the Computer Science & Information Technology (CS & IT), Copenhagen, Denmark, 25–26 January 2025; Volume 15, pp. 115–134. <https://doi.org/10.5121/csit.2024.150208>.
48. Zhao, S.; Tuan, L.A.; Fu, J.; Wen, J.; Luo, W. Exploring Clean Label Backdoor Attacks and Defense in Language Models. *IEEE/ACM Trans. Audio, Speech, Lang. Process.* **2024**, *32*, 3014–3024. <https://doi.org/10.1109/TASLP.2024.3407571>.
49. Oliveira, D.; Liu, J.; Pacitti, E. *Data-Intensive Workflow Management*; Springer: Berlin/Heidelberg, Germany, 2019.
50. Brooks, N.; Vance, C.; Ames, D. Cloud Computing: A Review of Evolution, Challenges, and Emerging Trends. *J. Comput. Sci. Softw. Appl.* **2025**, *5*, 1–17. <https://doi.org/10.5281/zenodo.15165068>.
51. Jain, P.; Banerjee, J. A Comparative Analysis of Cloud-Native, Cloud- Enabled, and Cloud-Agnostic Digital Transformation. *J. Emerg. Technol. Innov. Res.* **2024**, *11*, a463.

52. Barron-Lugo, J.A.; Gonzalez-Compean, J.; Lopez-Arevalo, I.; Carretero, J.; Martinez-Rodriguez, J.L. Xel: A Cloud-Agnostic Data Platform for the Design-Driven Building of High-Availability Data Science Services. *Future Gener. Comput. Syst.* **2023**, *145*, 87–103. <https://doi.org/10.1016/j.future.2023.03.019>.
53. Authors, T.K. Kubernetes—Production-Grade Container Orchestration. 2025. Available online: <https://kubernetes.io/> (accessed on 31 July 2026).
54. Donovan, R.; Healy, M.; Zheng, H.; Engel, F.; Vu, B.; Fuchs, M.; Walsh, P.; Hemmje, M.; Mc Kevitt, P. SenseCare: Using Automatic Emotional Analysis to Provide Effective Tools for Supporting. In Proceedings of the 2018 IEEE International Conference on Bioinformatics and Biomedicine, Madrid, Spain, 3–6 December 2018; pp. 2682–2687.
55. Inngest. Inngest Documentation. 2025. Available online: <https://www.inngest.com/docs> (accessed on 31 July 2026).
56. Norman, D.A.; Draper, S.W. *User Centered System Design; New Perspectives on Human-Computer Interaction*; L. Erlbaum Associates Inc.: Mahwah, NJ, USA, 1986.
57. Jacobson, L.; Booch, J.R.G. *The Unified Modeling Language Reference Manual*; Addison-Wesley: Boston, MA, USA, 1999.
58. Freund, F.; Tamla, P.; Wilde, F.; Hemmje, M. A FIT4NER Generic Approach for Framework-Independent Medical Named Entity Recognition. *Information* **2025**, *16*, 554. <https://doi.org/10.3390/info16070554>.
59. PocketBase. Pocketbase: Open Source Backend in 1 File. Version 0.32.0. 2025. Available online: <https://pocketbase.io/> (accessed on 31 July 2026).
60. BiocXml. BioC: XML/JSON Format Can Be Used to Share Text Documents and Annotations. 2025. Available online: <https://bioc.readthedocs.io/en/latest/> (accessed on 31 July 2026).
61. Polson, P.G.; Lewis, C.; Rieman, J.; Wharton, C. Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *Int. J. Man-Mach. Stud.* **1992**, *36*, 741–773. [https://doi.org/10.1016/0020-7373\(92\)90039-N](https://doi.org/10.1016/0020-7373(92)90039-N).
62. *IEEE Std 1028-2008*; IEEE Standard for Software Reviews and Audits. IEEE: New York, NY, USA, 2008. <https://doi.org/10.1109/IEEESTD.2008.4601584>.
63. Kasunic, M. *Designing an Effective Survey*; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2005.
64. Freund, F.; Tamla, P.; Tran, B.; Hemmje, M. Evaluating NERFlow: User-Centered Assessment of Automated LLM-Based Annotation for Medical Named-Entity Recognition. *Procedia Comput. Sci.* **2026**, *280*, 1350–1355. <https://doi.org/10.1016/j.procs.2026.04.183>.
65. Cohen, K.B.; Verspoor, K.; Fort, K.; Funk, C.; Bada, M.; Palmer, M.; Hunter, L.E. The Colorado Richly Annotated Full Text (CRAFT) Corpus: Multi-Model Annotation in the Biomedical Domain. In *Handbook of Linguistic Annotation*; Ide, N., Pustejovsky, J., Eds.; Springer: Dordrecht, The Netherlands, 2017; pp. 1379–1394. https://doi.org/10.1007/978-94-024-0881-2_53.
66. Monarch Initiative. Mondo Disease Ontology. 2025. Available online: <https://mondo.monarchinitiative.org/> (accessed on 31 July 2026).
67. Ollama. Ollama. 2025. Available online: <https://ollama.com/> (accessed on 31 July 2026).
68. Sun, S.; Hu, Q.; Xu, F.; Hu, F.; Wu, Y.; Wang, B. Medical Named Entity Recognition Based on Domain Knowledge and Position Encoding. *BMC Med. Inform. Decis. Mak.* **2025**, *25*, 235. <https://doi.org/10.1186/s12911-025-03037-0>.
69. Keraghel, I.; Morbieu, S.; Nadif, M. Recent Advances in Named Entity Recognition: A Comprehensive Survey and Comparative Study. *arXiv* **2024**, arXiv:2401.10825. <https://doi.org/10.48550/arXiv.2401.10825>.
70. NaturalNode. Natural: General Natural Language Facility for Node.js. Version 1.2.3. 2025. Available online: <https://github.com/NaturalNode/natural> (accessed on 31 July 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.