computation

MDPI

Article

SPE-ECG: An Ontology-Guided Knowledge Representation Framework for Enhancing LLM-Based Code Generation

Talha Mahmood [1,†], Muhammad Tanvir Afzal [1,2,9,†], Ahmad Sami Al-Shamayleh [3], Ghulam Mustafa [1,*,†], Isabel De la Torre Díez [4], Lázaro Javier Hernández Rodríguez [5,6,7] and Pablo Herrero García [5,8,10]

1 Faculty of Computing, Shifa Tameer-e-Millat University, Islamabad 44000, Pakistan; talha.ssc@stmu.edu.pk (T.M.); tanvir.afzal@gisma.com (M.T.A.)
2 Department of Computing and Data Science, GISMA University of Applied Sciences, 14469 Potsdam, Germany
3 Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Al-Ahliyya Amman University, Amman 19111, Jordan; a.alshamayleh@ammanu.edu.jo
4 eHealth and Telemedicine Group, University of Valladolid, 47011 Valladolid, Spain; isator@uva.es
5 Departamento de Soluciones Tecnológicas y Sistemas, Escuela Politécnica Superior, Universidad Europea del Atlántico, Isabel Torres 21, 39011 Santander, Spain; lazaro.hernandez@uneatlantico.es (L.J.H.R.); pablo.herrero@uneatlantico.es (P.H.G.)
6 Departamento de Ciencias de la Computación, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
7 Department of Computer Science, Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
8 Departamento de Ciências da Computação, Universidade Internacional do Cuanza, Estrada Nacional 250, Bairro Kaluapanda, Cuito, Bié, Angola
9 Department of Computer Science, Gulf University for Science and Technology (GUST), P.O. Box 7207, Hawally 32093, Kuwait
10 Departamento de Ciencias de la Computación, Universidad de La Romana, La Romana 22000, Dominican Republic
* Correspondence: ghulam.mustafa.ssc@stmu.edu.pk
† These authors contributed equally to this work.

Abstract

Large Language Models (LLMs) can translate natural-language instructions into code, but when the input is unstructured clinical protocol text, the generated code may omit doses, monitoring intervals, or safety thresholds that the text states. Such omissions matter because they can propagate into software that supports clinical monitoring. We propose SPE-ECG (Structured Prompt Engineering for Enhanced Code Generation), a transparent, rule-based pipeline that converts unstructured domain text into a JSON prompt using text cleaning, part-of-speech tagging, bigram matching, and hand-built domain dictionaries, without fine-tuning any model. In a case study of 43 rheumatoid arthritis DMARD monitoring protocols, we compared raw and structured prompts on three LLMs (GPT-3.5, LLaMA 2, and Gemma), scoring the generated Python code using version 3.15.1 with BLEU against the raw-prompt code of the other two models. Mean BLEU rose for all three models (relative gains of 55.1%, 32.5%, and 19.9%); the paired differences were significant for GPT-3.5 and LLaMA 2 but not, after correction for multiple comparisons, for Gemma. These results suggest that lightweight input structuring can change how closely different models' outputs agree, and that it could serve as a low-cost, auditable preparation step for protocol-to-code tools built on free or open models. BLEU does not measure correctness, no ablation or state-of-the-art baseline was run, and clinical validation is required before any use.

Keywords: large language models; structured prompt engineering; ontology-guided prompting; code generation; knowledge representation; intelligent systems

Check for updates

Received: 9 September 2026
Revised: 26 October 2026
Accepted: 27 October 2026
Published: 3 November 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) license.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in translating natural-language instructions into executable code, enabling new forms of intelligent automation in software engineering, healthcare informatics, financial systems, and legal technology [1–3]. By leveraging transformer-based architectures trained on large-scale corpora, these models can synthesize syntactically valid code from free-form textual descriptions. Despite these advances, reliable code generation in knowledge-intensive domains remains a challenging problem [4–6].

In real-world applications, input prompts are rarely clean or fully specified. Domain-specific text often contains ambiguous phrasing, implicit procedural dependencies, specialized terminology, and incomplete parameter definitions. When such unstructured inputs are directly supplied to LLMs, models may produce code that is syntactically correct but incomplete with respect to the domain requirements stated in the text. This limitation is particularly critical in high-stakes domains—such as healthcare, finance, and legal informatics—where correctness, reproducibility, and procedural compliance are essential.

1.1. Terminology and Scope

Throughout this paper, *semantic alignment* refers to alignment with the functional requirements stated in the source text: whether the generated code represents the domain entities (e.g., drug names), parameter values (e.g., doses and intervals), temporal rules (e.g., monitoring frequencies), and conditional or safety constraints (e.g., escalation criteria) that the text specifies. It does not refer to the naming of variables, methods, or classes, and it is not a claim about general software quality. The artefacts studied are short, self-contained Python scripts that implement a single dosing or monitoring protocol The libraries used by the generated code were not recorded. No application architecture, frameworks, persistence layers, or user interfaces are generated, and software-engineering properties such as maintainability, complexity, test coverage, and performance are outside the scope of this study.

Existing approaches attempt to mitigate this issue through data preprocessing, contextual embeddings, instruction tuning, or domain-specific fine-tuning [7–9]. While these techniques improve linguistic representation, they do not explicitly encode structured domain knowledge at the prompt level. Structured prompt engineering has emerged as a promising alternative, organizing inputs into more interpretable formats such as JSON or XML [10]. However, current approaches typically focus on syntactic restructuring rather than systematic semantic alignment through ontology-guided knowledge representation. As a result, the integration of formal domain semantics into prompt engineering remains underexplored.

1.2. Problem and Gap

The specific problem addressed here is the *omission and distortion of protocol details* when an LLM turns semi-structured clinical protocol text into code. Protocol text packs doses, monitoring intervals, laboratory thresholds, and escalation actions into long table cells, and nothing in a raw prompt tells the model which parts are the entities, the thresholds, or the actions; in our outputs, multi-stage instructions were often simplified. Existing remedies either change the model (fine-tuning or domain pretraining, which needs data and computing resources) or change the output format (structured or reasoning-style prompting, which does not tell the model what the domain-specific elements of the input are). The gap we address is the lack of a transparent, low-cost, model-agnostic methods to make the domain-specific elements of the input explicit before generation, together with the lack of evidence on whether doing so changes the code that different LLMs produce. We

address it with SPE-ECG and evaluate it in one case study, without claiming to outperform fine-tuned or code-specialized models.

This work addresses the following research question:

Can ontology-guided structured prompt engineering change the lexical agreement (BLEU) of LLM-generated code with the code that other LLMs generate from raw-text prompts, relative to raw-text prompts and without model fine-tuning?

The primary metric (BLEU) measures n-gram overlap with the raw-prompt code of other models; no independent reference implementation was used, and BLEU measures neither functional correctness nor clinical safety. The study therefore addresses lexical agreement between models only, and the implications for reliability are discussed as limitations (Section 5).

To answer this question, we propose **SPE-ECG** (**Structured Prompt Engineering for Enhanced Code Generation**), a model-agnostic framework that transforms unstructured domain text into a standardized JSON schema using rule-based linguistic processing, curated domain dictionaries, and a lightweight domain ontology. The framework extracts task definitions, entities, parameters, and procedural constraints and encodes them explicitly in a machine-interpretable structure, with the aim of reducing ambiguity at the prompt level.

The framework is evaluated on three general-purpose LLMs—GPT-3.5, LLaMA 2, and Gemma—under both unstructured and structured prompting conditions. Mean BLEU was higher under structured prompting for all three models, with relative gains of 55.1% (GPT-3.5), 32.5% (LLaMA 2), and 19.9% (Gemma). A healthcare case study involving Rheumatoid Arthritis (RA) treatment protocols [11,12] serves as a knowledge-intensive testbed. The gains were statistically significant for GPT-3.5 and LLaMA 2 but not for Gemma after correction for multiple comparisons. The study evaluates a single domain: the framework is designed so that its dictionaries and ontology can be replaced to target other domains, but this has not been tested here.

The contributions of this work are summarized as follows:

- We introduce **SPE-ECG**, a structured prompt engineering framework that uses curated dictionaries and a lightweight domain ontology to improve the alignment of LLM-generated code with domain requirements.
- We design a transparent, rule-based JSON structuring pipeline that combines linguistic preprocessing with dictionary- and ontology-based field assignment, and we specify its rules and construction procedure in detail.
- We provide an empirical comparison of raw and structured prompts across three LLMs on an RA/DMARD case study, reporting BLEU-based lexical agreement between models with paired statistics, and we state explicitly what BLEU does and does not establish.
- We discuss input-level knowledge structuring as a lightweight alternative to fine-tuning, together with the validity threats (single domain, lexical-overlap metric, choice of references, confounded comparison) that must be addressed before stronger claims can be made.

Section 2 reviews related work. Section 3 introduces the SPE-ECG framework and methodology. Section 4 presents experimental results. Section 5 discusses implications, and Section 6 concludes the study.

2. Literature Review

Research on automated code generation has evolved substantially with the advancement of Artificial Intelligence (AI), Natural Language Processing (NLP), and transformer-based architectures. Early systems relied on symbolic and rule-based approaches that translated structured inputs into executable programs through predefined grammars and

logic templates [13]. These systems provided interpretability and deterministic outputs; however, they required manually engineered rules and lacked robustness when processing unstructured or ambiguous natural language [14]. Consequently, rule-based paradigms struggled to scale to complex, domain-rich environments.

The introduction of the transformer architecture [15] marked a paradigm shift in automated code generation. Models such as GPT, Codex, and LLaMA demonstrated the ability to synthesize syntactically correct code from free-form textual prompts by leveraging large-scale pretraining on diverse corpora [2,16,17]. These models capture long-range dependencies and contextual relationships, enabling zero-shot and few-shot code synthesis. Despite their strong generative capacity, empirical evaluations have shown that transformer-based models can be sensitive to prompt phrasing, underspecified parameters, and implicit procedural dependencies [6]. In knowledge-intensive domains, this sensitivity may result in incomplete logic, omitted constraints, or semantically misaligned outputs.

To mitigate these limitations, researchers have explored instruction tuning and embedding-based adaptation. Architectures such as T5 recast diverse language tasks in a unified text-to-text format [9], and multitask prompted training has been shown to improve zero-shot task generalization [7]. Domain-specific variants—including BioBERT and ClinicalBERT—enhanced terminology recognition in biomedical and clinical contexts through continued pretraining on specialized corpora [8,18]. While these approaches improve domain comprehension, they require significant computational resources and dataset availability. Moreover, they modify internal model representations rather than explicitly structuring the input knowledge supplied to the model.

Parallel to model-centric optimization strategies, structured prompt engineering has emerged as an input-centric alternative. Recent studies have examined how constraining prompts and outputs through formatted templates or structured schemas can influence reasoning performance and output reliability [10]. Format restrictions, including JSON- or XML-based formats, have been studied as a way to improve task alignment in natural language generation tasks. However, most structured prompting approaches focus primarily on syntactic organization rather than semantic grounding. Specifically, they rarely incorporate formal ontological representations to ensure consistent encoding of domain entities, parameters, and procedural constraints.

A related line of work structures the *reasoning* that precedes code generation. Structured chain-of-thought prompting asks the model to produce intermediate reasoning organized by program structures (sequence, branch, and loop) before writing code [19]; planning-based prompting for code generation follows a similar idea [2]. SPE-ECG differs in that it structures the *input* using domain dictionaries rather than eliciting structured reasoning from the model; the two ideas are complementary and are not compared experimentally in this paper. Knowledge-graph- and ontology-augmented prompting likewise injects external domain knowledge into LLM inputs [20]; such approaches typically rely on large knowledge resources, whereas SPE-ECG uses a small, hand-curated schema.

Ontology-driven knowledge representation has a long-standing tradition in intelligent systems research. Ontologies enable explicit modeling of domain concepts, relationships, and constraints, supporting semantic interoperability and reasoning [21,22]. In domains such as healthcare and finance, ontological modeling has been used to support decision support systems and compliance automation. Yet, the integration of ontological structures directly into prompt engineering for LLM-based code generation remains underexplored.

In clinical informatics, computer-interpretable guideline formalisms such as GLIF [23], PROforma [24], and Arden Syntax [25] represent guideline logic in executable form. These formalisms require expert authoring of formal models and were not designed as prompt

formats for LLMs; SPE-ECG instead extracts a flat structured representation from free text and leaves code generation to an LLM.

Finally, the evaluation of generated code ranges from n-gram similarity metrics such as BLEU [26] and its code-aware variant CodeBLEU [27], which adds syntax-tree and data-flow matching, to execution-based metrics such as pass@k [16,28]. BLEU is inexpensive and reproducible but measures lexical overlap only; this limits the conclusions that can be drawn from it (Section 5.7).

A comparative summary of representative approaches is provided in Table 1. The analysis indicates that existing paradigms either (i) rely on deterministic but inflexible rule-based systems, (ii) depend on large-scale pretrained transformers without structured semantic guidance, or (iii) require computationally intensive domain-specific fine-tuning. Structured prompting studies demonstrate the influence of format constraints but do not systematically integrate ontology-guided semantic extraction for executable code generation tasks.

Table 1. Comparative analysis of representative approaches in automated code generation and structured prompting.

Study	Approach	Focus	Strengths	Limitations
Gulwani et al. [13]	Rule- and search-based synthesis	Program synthesis	Deterministic logic	Limited scalability
Liu et al. [14]	Rule-based IE	Open text extraction	Transparent rules	Poor adaptability
GPT-3.5 (ChatGPT) [29]	Transformer LLM	General code generation	Strong zero-shot capability	Sensitive to prompt ambiguity
Codex [16]	Transformer LLM	Software engineering	High syntactic correctness	Limited domain structuring
StructCoder [30]	Structure-aware transformer	Code tasks	Incorporates structural bias	No external ontology integration
T5 [9]	Text-to-text pretrained model	Multi-task learning	Unified task format	Requires fine-tuning
BioBERT [8]	Domain-adaptive model	Biomedical NLP	Terminology specialization	Not optimized for code generation
ClinicalBERT [18]	Domain-adaptive model	Clinical NLP	Domain-aware embeddings	Computational cost
OpenMedLM [31]	Prompt optimization	Medical QA	Enhanced prompt design	Limited executable evaluation
Tam et al. [10]	Format-constrained prompting	NLP reasoning	Reduced ambiguity	No ontology-based semantic structuring
Li et al. [19]	Structured chain-of-thought prompting	Code generation	Structured reasoning without fine-tuning	Model-generated structure; no domain ontology
GLIF, PROforma, Arden [23–25]	Guideline formalisms	Clinical guidelines	Formal, executable representation	Manual expert authoring; not LLM prompt formats
SPE-ECG (Proposed)	Dictionary- and ontology-guided structured prompting	Code generation from domain text (one domain evaluated)	Explicit field-level encoding; no fine-tuning	Rule/dictionary-based; flat schema; single-domain evaluation

Research Gap

The reviewed literature reveals a methodological gap. Existing model-centric approaches focus on improving internal representations through scaling or fine-tuning, whereas structured prompting studies emphasize format constraints without formal semantic grounding. Among the works we surveyed, we did not identify one that systematically combines ontology-based knowledge extraction with structured prompt generation for executable code synthesis in knowledge-intensive domains; this is a statement about the surveyed literature and not an exhaustive claim.

This gap motivates the proposed SPE-ECG framework, which operates at the input representation level by transforming unstructured domain text into a dictionary- and ontology-aligned JSON schema. By combining rule-based linguistic processing with a lightweight domain model, SPE-ECG aims to improve the alignment of generated code with domain requirements without requiring modifications to underlying model architectures. Specifically, among the works we surveyed we did not find one that (i) converts unstructured clinical protocol text into a fixed key–value schema using a small hand-built dictionary and (ii) compares the code that several LLMs generate from raw and structured versions of the same text. SPE-ECG targets this gap; it is a modest, auditable step and not a claim of state-of-the-art performance.

3. Methodology

This section presents the methodological framework underlying the proposed **Structured Prompt Engineering for Enhanced Code Generation (SPE-ECG)** approach. The primary objective of SPE-ECG is to improve the alignment between domain text and generated code—in terms of the entities, parameters, and constraints that the text specifies—by transforming unstructured domain text into a structured JSON representation guided by curated dictionaries and a lightweight domain ontology.

In conventional prompting, domain-specific instructions are provided to Large Language Models (LLMs) in free-text form. Such inputs often contain implicit constraints, ambiguous phrasing, incomplete parameter definitions, and domain-specific terminology that may not be consistently interpreted by the model. As a result, generated code can be syntactically valid yet incomplete with respect to the requirements stated in the text. SPE-ECG addresses this limitation by introducing an intermediate structuring layer that explicitly encodes domain knowledge prior to model inference.

The proposed framework converts unstructured textual input into a standardized JSON schema through dictionary- and ontology-based extraction and rule-based linguistic processing. This structured representation explicitly separates task definitions, domain entities, parameters, and procedural constraints. The framework is designed to be model-agnostic, and its dictionaries and ontology can in principle be replaced to target other domains; in this study, however, it is instantiated and evaluated only on the RA/DMARD case study.

Figure 1 illustrates the overall SPE-ECG pipeline. The methodology comprises four interconnected stages: (i) prompt engineering and model selection, (ii) ontology development and semantic extraction, (iii) structured prompt generation and code synthesis, and (iv) comparative evaluation. Each stage is described in detail in the following subsections.

3.1. Part 1: Prompt Engineering and Model Selection

The first stage of the SPE-ECG framework establishes the experimental foundation by defining the domain corpus, prompt construction strategy, and evaluation conditions across multiple Large Language Models (LLMs). The objective is to compare raw and structured prompting under otherwise identical conditions across model architectures.

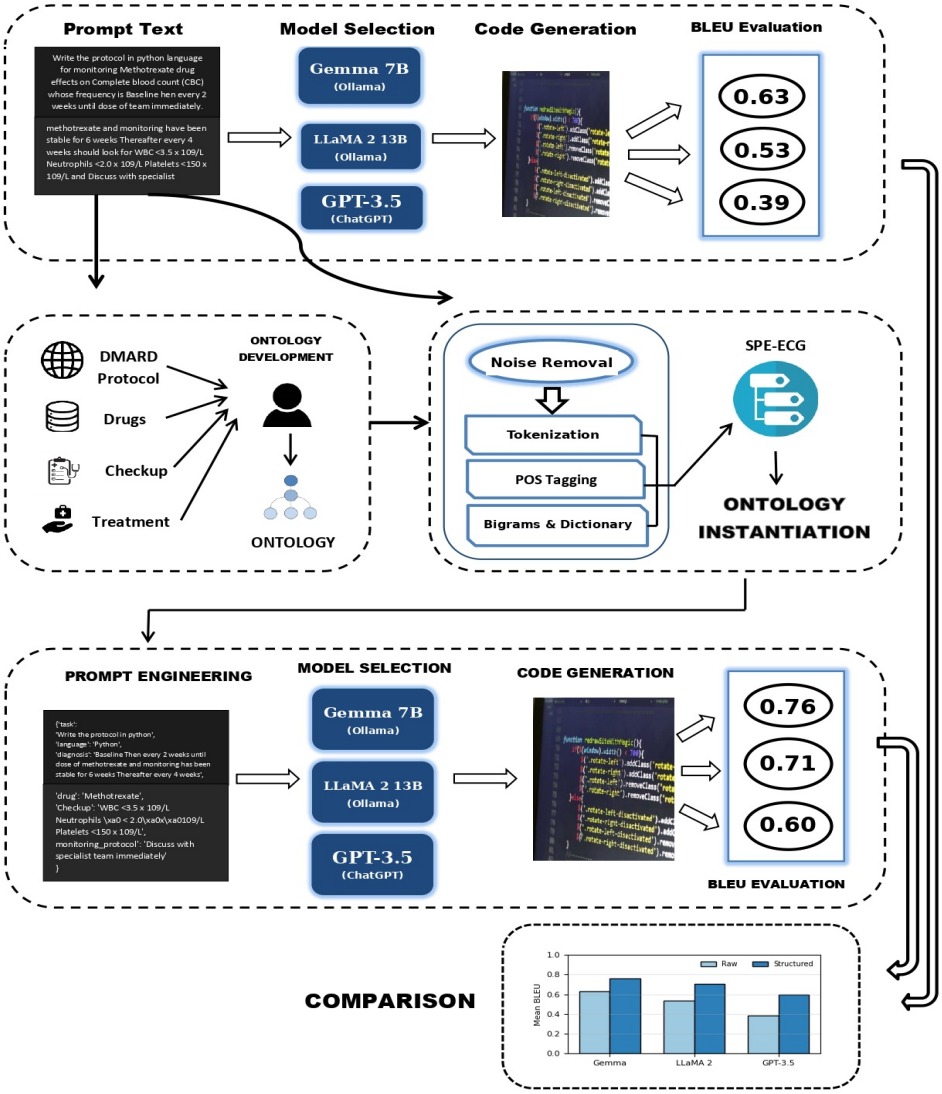


Figure 1. Proposed methodology pipeline for the SPE-ECG framework. The four stages integrate ontology development, structured prompt design, and comparative evaluation of LLMs.

3.1.1. Dataset Definition and Data Collection

Let the domain-specific corpus be defined as:

$$\mathcal{D} = \{T_1, T_2, \ldots, T_N\} \tag{1}$$

where each document T_i represents an unstructured textual description of a domain protocol. In this study, $N = 43$ protocols ("documents") extracted from a single web page and corresponding to Rheumatoid Arthritis (RA) treatment guidance involving Disease-Modifying Antirheumatic Drugs (DMARDs).

The corpus was collected from a single publicly accessible web page, the bpac^{nz} article on DMARD monitoring (https://bpac.org.nz/BPJ/2008/October/dmards.aspx (accessed on 1 May 2024)), from which 43 DMARD monitoring protocols were extracted by web crawling; each protocol is one source text T_i. The page is a 2008 New Zealand clinical guidance article, so the corpus has a single source. The page was accessed in 2024 and is publicly available; no separate permission for reuse was obtained. The protocols are used in clinical practice by the doctors who suggested them as study material (early 2024); no clinician reviewed the dictionaries, the prompts, or the generated code. Each protocol contains pharmacological directives, dosage schedules, monitoring requirements, and

conditional treatment procedures. Figure 2 illustrates the overall data acquisition and preprocessing workflow.

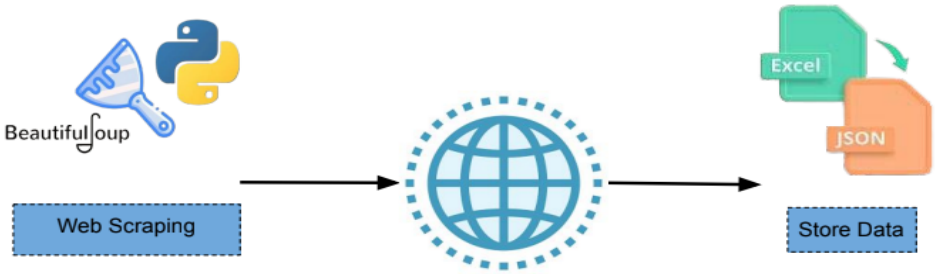


Figure 2. Data acquisition and preprocessing workflow for the SPE-ECG framework. Domain-specific documents are collected, cleaned, and prepared for ontology-guided semantic extraction.

Each protocol T_i yields a set of prompts:

$$\mathcal{P}_i = \{p_{i1}, p_{i2}, \ldots, p_{ik}\} \quad (2)$$

where each prompt encodes a specific task such as monitoring drug dosage, scheduling tests, or applying conditional logic. In this study, each protocol yields a single prompt ($k = 1$), so the number of evaluation prompts is $M = N = 43$, and the same 43 prompts are used for every model and condition. Each protocol is one row of the table on the page, and the prompt for that row was built by the authors by concatenating the sentences of the row. All 43 rows were used. Each raw prompt consists of the instruction "write the protocol in python language", the drug name taken from the table, and the text of the row, in which the content of each cell is preceded by the header of that cell. The text is used as extracted, without rewording.

Table 2 summarizes the compiled corpus and the dictionary categories used for structured extraction. The entity, monitoring, frequency, and condition-specific entries are the dictionary terms compiled by the authors from the corpus (Section 3.4.2); they are not an external resource. The number of DMARD mentions (43) equals the number of protocols because each protocol mentions its drug once.

Table 2. Statistical summary of the RA case study corpus and dictionary categories used for ontology-guided extraction. Counts are total occurrences in the text, not numbers of distinct dictionary entries.

Feature	Description	Count
DMARD Entities	DMARD mentions in the protocols	43
Monitoring Keywords	Clinical safety and validation terms	84
Frequency Keywords	Scheduling and temporal indicators	139
Condition-Specific Terms	Disease progression and assessment indicators	31
Total Protocols	Number of protocols (documents) in corpus	43

The dictionary categories in Table 2 form the basis for ontology instantiation and rule-based extraction described in subsequent sections.

3.1.2. Prompting Conditions

To evaluate the impact of structured representation, two prompting conditions are defined:

$$\mathcal{P}^{raw} = \{p_1^{raw}, p_2^{raw}, \ldots, p_M^{raw}\} \quad (3)$$

$$\mathcal{P}^{struct} = \{\Phi(p_1^{raw}), \Phi(p_2^{raw}), \ldots, \Phi(p_M^{raw})\} \quad (4)$$

where:

- $\mathcal{D}^{raw}$ denotes the original unstructured prompts;
- $\Phi(\cdot)$ represents the SPE-ECG transformation function;
- $\mathcal{D}^{struct}$ denotes ontology-guided structured prompts;
- M is the total number of evaluation prompts.

The raw prompt p_k^{raw} consists of the instruction "write the protocol in python language", the drug name taken from the table, and the text of the table row, in which the content of each cell is preceded by the cell's header. The structured prompt $\Phi(p_k^{raw})$ is the JSON object obtained from the same text (Section Worked Example) and is submitted *without* any added instruction. The two conditions therefore differ in more than structure: the raw prompt carries an explicit instruction and the table headers, whereas the structured prompt carries the task only through the content of its `task` and `language` fields. In addition, because Φ retains only content that matches the dictionaries, the structured prompt is not informationally equivalent to the raw prompt (unmatched content is dropped; see Section 3.4.3). Observed differences therefore reflect the combined effect of reformatting, dictionary-based selection of content, and the different wording, not representation format alone. The fraction of prompt tokens retained by Φ (coverage) was not measured in this study and is noted as a limitation.

3.1.3. Model Selection and Experimental Setup

Three LLMs are selected for their accessibility: one model available through a free web interface and two open-weight models of different sizes that were run locally [17,29,32]:

$$\mathcal{M} = \{M_{GPT3.5}, M_{LLaMA2}, M_{Gemma}\} \tag{5}$$

For each model $M_j \in \mathcal{M}$, code outputs are generated under both prompting conditions:

$$C_j^{raw} = M_j(p^{raw}) \tag{6}$$

$$C_j^{struct} = M_j(\Phi(p^{raw})) \tag{7}$$

The experimental objective is to evaluate whether structured prompting improves performance:

$$\mathbb{E}[Score(C^{struct})] > \mathbb{E}[Score(C^{raw})] \tag{8}$$

where $Score(\cdot)$ denotes the evaluation metric (BLEU in this study), and $\mathbb{E}[\cdot]$ represents expectation over all prompts.

All three are general-purpose, instruction-tuned models; none is a code-specialized variant (such as Code Llama or CodeGemma). The selection is a convenience sample motivated by accessibility (a free web interface and models that can be run locally with Ollama), and it does not include current frontier or code-specific models (Section 5.7). The comparison is formalised as a paired hypothesis over the M prompts, tested as described in Section 3.3.1; the formal protocol, including exact model identifiers and inference settings, is given in Section 3.3.2.

3.2. Data Preprocessing

Following data acquisition, a preprocessing stage is applied to standardize and refine the corpus prior to semantic extraction. The purpose of this stage is to remove irrelevant artifacts while preserving domain-critical terminology and procedural logic. Let the raw corpus be denoted as:

$$\mathcal{D}^{raw} = \{T_1^{raw}, T_2^{raw}, \ldots, T_N^{raw}\} \tag{9}$$

The preprocessing function $\Psi(\cdot)$ transforms the raw corpus into a cleaned version:

$$\mathcal{D}^{clean} = \Psi(\mathcal{D}^{raw}) \tag{10}$$

This transformation ensures that downstream ontology-guided structuring operates on semantically coherent and noise-free text.

3.2.1. Noise Removal

Raw documents extracted from online repositories frequently contain non-semantic artifacts such as HTML tags, encoding symbols, duplicated headers, and extraneous metadata. These elements introduce variability that may negatively affect tokenization and semantic extraction.

The scraped text was cleaned with the BeautifulSoup library, which was used to remove:

- HTML and markup tags,
- Stray characters and encoding symbols left over from scraping,
- Redundant whitespace and formatting artifacts,
- Irrelevant metadata fields.

Numbers and units in doses (for example "7.5 mg" or "10 mg/week") were kept exactly as they appear on the page. Noise removal acts only on markup and stray characters and does not filter vocabulary, so domain terminology is preserved. A term that is absent from the dictionaries will not be extracted later (Section 5.7), and some non-breaking-space characters remain in the extracted text (they are visible in the example structured prompt of Figure 1), which is a limitation of the cleaning step.

3.2.2. Tokenization and Linguistic Normalization

After cleaning, each document $T_i^{clean} \in \mathcal{D}^{clean}$ is decomposed into a sequence of tokens:

$$T_i^{clean} = \{w_1, w_2, \ldots, w_m\} \tag{11}$$

Tokenization is implemented using the `Punkt` tokenizer from the NLTK library. Sentence-level and word-level segmentation enables subsequent identification of domain entities, constraints, and scheduling expressions.

Where necessary, additional normalization steps are applied, including:

- Lowercasing (when not semantically sensitive),
- Stop-word filtering (excluding domain-relevant functional words),
- Preservation of numerical dosage and frequency indicators.

Unlike aggressive text simplification techniques, preprocessing is deliberately conservative to avoid removing semantically meaningful procedural expressions.

3.2.3. Structured Storage and Corpus Representation

The processed corpus is stored in tabular format, where each row corresponds to one protocol, i.e., one row of the table on the source page. Formally, the prompt-ready corpus is represented as:

$$\mathcal{R} = \{r_1, r_2, \ldots, r_M\} \tag{12}$$

where each record r_k stores the document identifier, the cleaned text of one instruction segment, and, after the structuring stage, the fields populated by Φ for that segment (drug, diagnosis, checkup values, and monitoring protocol). These components are not derived from isolated tokens: they are obtained by matching the bigrams of the segment against the dictionaries (Section 3.4.3), so a field is populated only when a dictionary entry matches. This intermediate representation serves as the input to the ontology-guided transformation stage described in the next subsection. The corpus is stored in Excel files.

3.3. Code Evaluation Using BLEU Metric

To assess the impact of ontology-guided structured prompting, generated code outputs are scored with the Bilingual Evaluation Understudy (BLEU) metric [26]. BLEU measures the lexical similarity between a candidate and reference scripts based on n-gram overlap; it does not execute the code. Because no independent reference implementation exists for the protocols, the code generated by each model is scored against the raw-prompt code generated by the other two models.

Let $C_{j,k}$ denote the code generated by model M_j for prompt p_k, and let $\mathcal{R}_{j,k} = \{C^{raw}_{j',k} : j' \neq j\}$ denote the set of two reference codes, namely, the raw-prompt code generated by the other two models for the same prompt. The BLEU score for each instance is computed with both references as:

$$\mathrm{BLEU}(C_{j,k}, \mathcal{R}_{j,k}) = \mathrm{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{13}$$

where:

- BP is the brevity penalty;
- p_n represents modified n-gram precision;
- N is the maximum n-gram order (set to 4);
- w_n are uniform weights such that $\sum w_n = 1$.

BLEU is bounded in $[0, 1]$ and all values in this paper are reported on that scale. BLEU was computed with the NLTK library using its default settings (no smoothing function was specified), with two references per candidate; all per-prompt scores lie in $[0, 1]$.

For each model M_j, the overall performance under a given prompting condition is defined as the mean BLEU score across all evaluation prompts:

$$\overline{BLEU}_j = \frac{1}{M} \sum_{k=1}^{M} \mathrm{BLEU}(C_{j,k}, \mathcal{R}_{j,k}) \tag{14}$$

Relative improvement between structured and unstructured prompting is computed as:

$$\Delta_j = \frac{\overline{BLEU}^{struct}_j - \overline{BLEU}^{raw}_j}{\overline{BLEU}^{raw}_j} \times 100\% \tag{15}$$

This formulation expresses the change in the mean score independent of model architecture. Because it is a ratio of means, it is sensitive to the size of the baseline mean and is reported together with the paired per-prompt analysis below.

3.3.1. Statistical Analysis

Every prompt p_k is scored under both conditions, so the conditions form paired samples. For each model, we report the number of evaluation prompts M; the mean, standard deviation, median, and interquartile range of per-prompt BLEU under each condition, as well as the mean paired difference $d_{j,k} = \mathrm{BLEU}^{struct}_{j,k} - \mathrm{BLEU}^{raw}_{j,k}$ with a 95% percentile-bootstrap confidence interval (10,000 resamples of prompts). The primary test is a two-sided Wilcoxon signed-rank test on $d_{j,k}$ for each model, with Holm correction across the three models; the effect size is the rank-biserial correlation. We also report the proportion of prompts for which structured prompting scored higher than raw prompting. One generation was produced per prompt, condition, and model, so prompts are the unit of analysis. The word "significant" is used only for the outcome of these tests. As an exploratory secondary analysis, the dispersion of per-prompt BLEU is compared descriptively between conditions (standard deviation and interquartile range); no inferential claim about variance is made.

3.3.2. Experimental Configuration

The configuration of each model is given in Table 3. The default settings of each model's interface or runtime were used and were not modified, so sampling (e.g., temperature) was not controlled, and one generation was produced per prompt and condition. The same defaults apply to the raw and structured conditions, and no few-shot examples are provided.

Table 3. Experimental Details of LLMs.

Model	Version	Access Route	Temperature, Top-p	Max. New Tokens	Generations per Prompt
GPT-3.5	GPT-3.5 (free ChatGPT, June–July 2024)	ChatGPT web interface	Default (not modified)	Default	1
LLaMA 2 (13B)	LLaMA 2, 13B parameters	Ollama (local)	Default (not modified)	Default	1
Gemma (7B)	Gemma, 7B parameters	Ollama (local)	Default (not modified)	Default	1

Prompt delivery. Each raw prompt was submitted as written (Section 3.1.2); each structured prompt was submitted as the JSON object alone, with no added instruction. All 43 prompts were submitted in a single chat per model (a single ChatGPT conversation, and a single Ollama session for each of LLaMA 2 and Gemma), so earlier prompts and answers remained in the model's context and may have influenced later outputs.

Reference code. No independent reference implementation was used. For each prompt, the code generated by a model was scored against the raw-prompt code of the other two models (two references per candidate; for example, GPT-3.5 code against the LLaMA 2 and Gemma raw-prompt code). The same raw-prompt references are used for the raw and the structured conditions. BLEU therefore measures lexical agreement with other models' raw-prompt output and not correctness: a structured-prompt output scores highly when it resembles what the other models write from raw prompts, which is not by itself evidence of correct code. This is discussed in Section 5.7.

3.3.3. Scope of the Comparison

Only two conditions are compared: raw prompts and the complete SPE-ECG pipeline. Because the full pipeline changes several components at once (preprocessing, n-gram extraction, dictionaries, ontology-based assignment, and JSON structuring), the comparison cannot attribute any difference to ontology guidance specifically. Ablation and baseline conditions (plain JSON restructuring without dictionaries, extraction without ontology guidance, an explicit natural-language prompt, LLM-based extraction into the same schema, and few-shot prompting) were not evaluated and are listed as future work (Section 7). The hypothesis concerns the *input representation*: whether the same model, given the same protocol content, produces code that agrees more closely with other models' output when the content is supplied as structured JSON. The appropriate control for this hypothesis is the same model with the raw prompt, which is the comparison made here. Comparing SPE-ECG with state-of-the-art alternatives answers a different question, namely, whether it is better than other ways of improving code generation, for example, few-shot prompting, structured chain-of-thought prompting [19], LLM-based extraction into the same schema, fine-tuning, or newer and code-specialised models. Such baselines were not evaluated in this study, and we do not claim that SPE-ECG outperforms them.

3.3.4. Evaluation Considerations

While BLEU provides a standardized and reproducible measure of lexical similarity, it does not evaluate functional correctness or execution validity. BLEU scores are therefore indicators of surface-level agreement with the other models' raw-prompt code and not of semantic equivalence, reliability, or clinical safety. No execution-based or expert evaluation was performed, and BLEU (computed with NLTK) is the only metric used. CodeBLEU [27], execution- or unit-test-based correctness (pass@k [16]), and expert rating of whether each clinical constraint is implemented are listed as future work.

3.4. Textual to Structured Prompt Conversion

The central component of the SPE-ECG framework is the transformation of unstructured textual prompts into a semantically organized JSON representation. This process introduces an intermediate structured layer between natural-language input and model inference, thereby reducing ambiguity and improving semantic consistency.

Formally, let a preprocessed prompt be denoted as:

$$p_k \in \mathcal{D}^{raw} \tag{16}$$

The structured representation is obtained through the ontology-guided transformation function:

$$S_k = \Phi(p_k, O) \tag{17}$$

where:

- $\Phi(\cdot)$ denotes the rule-based extraction and structuring operator;
- $O = (E, R, A)$ represents the domain ontology;
- S_k is the resulting structured JSON object.

3.4.1. Ontology-Guided Schema Definition

The ontology O encodes domain knowledge as:

$$O = (E, R, A) \tag{18}$$

where:

- E: domain entities (e.g., drugs, financial instruments);
- R: relationships among entities (documented in the schema but not used by the current extraction rules; see below);
- A: attributes and procedural parameters.

Based on this ontology, the JSON schema used in this study has six fields:

$$S_k = \{\text{task, language, drug, diagnosis, Checkup, monitoring_protocol}\} \tag{19}$$

This schema enforces explicit separation of executable components prior to model invocation.

3.4.2. Ontology and Dictionary Construction

The ontology O used in this study is deliberately lightweight: a small domain-specific conceptual schema that groups the terms needed for the RA/DMARD case study into entity classes (E) and attribute or parameter classes (A), from which the dictionaries D_* are generated. It is not expressed in a formal ontology language (such as OWL or RDF) and is not aligned to an established terminology such as RxNorm, ATC, SNOMED CT, or UMLS. The relation set R (e.g., drug–test or drug–dose relations) is recorded in the schema but is *not used* by the extraction rules in the present implementation: matching depends

only on E and A through the dictionaries. "Ontology-guided" is therefore used here in a restricted sense: dictionary-based assignment of text spans to an entity–attribute schema.

Construction procedure. The dictionaries were built in five steps. (1) The cleaned text of the 43 protocol rows was read in full. (2) Candidate terms were listed field by field in a spreadsheet, following the inclusion rules in Table 4. (3) Variants of the same term (spelling, case, and unit notation) were merged into one entry. (4) Each entry was assigned to one field; a bigram that matches the dictionaries of two fields is not resolved by a priority rule but is discarded at extraction time (Section 3.4.3). (5) The final lists were stored as the dictionary files released in the repository. The work was done by hand by the authors and took about one week; no clinician or other independent expert reviewed the lists. The resulting numbers of entries are given in Table 2.

Table 4. Inclusion rules used to build the dictionary of each schema field, with examples from the methotrexate protocol.

Field	What Goes in the Dictionary	Example	Matched by
`task`	Instruction verbs that state what the code must do	"write"	Verbs (POS tag)
`language`	Names of the target programming language	"python"	Bigram lookup
`drug`	Names of the DMARDs in the protocol table	"methotrexate"	Nouns against the drug dictionary
`diagnosis`	Condition and schedule expressions that give the timing of monitoring	"every 2 weeks"	Bigram lookup
`Checkup`	Laboratory tests and their threshold expressions	"WBC < 3.5"	Bigram lookup
`monitoring_protocol`	Actions the clinician must take	"discuss with specialist"	Bigram lookup

Consistency. No reasoner or formal consistency checker was used, and no formal check of the dictionaries was performed beyond the authors' own reading. A bigram that matched the dictionaries of more than one field was discarded (Section 3.4.3), which avoided contradictory assignments at the cost of losing such bigrams.

Sensitivity to ontology quality. Extraction quality is bounded by dictionary coverage and correctness: a missing term is not extracted, and a mis-categorized term populates the wrong field. The present study does not quantify this sensitivity.

In the healthcare instantiation, the six schema fields are `task`, `language`, `drug`, `diagnosis`, `Checkup`, and `monitoring_protocol` (Section Worked Example).

3.4.3. Rule-Based Extraction Mechanism

The transformation function Φ operates over cleaned, tokenized, POS-tagged text. Its rules are dictionary-lookup rules: each dictionary entry maps a surface form to one schema field. The dictionaries are written by the authors (Section 3.4.2); the rules are neither learned nor generated automatically. Let $\mathcal{F} = \{\text{task}, \text{language}, \text{drug}, \text{diagnosis}, \text{Checkup}, \text{monitoring_protocol}\}$, and let D_f be the dictionary of field $f \in \mathcal{F}$, derived from the ontology. Extraction proceeds as follows. POS tags are assigned to the tokens; nouns are matched against the drug dictionary D_{drug}, and verbs fill the `task` field. In addition, a bag-of-words step is used to form bigrams (two-word sequences), and each bigram g is matched against the dictionaries; the set of fields whose dictionary contains g is

$$F(g) = \{f \in \mathcal{F} : g \in D_f\}. \tag{20}$$

If $|F(g)| = 1$, the bigram is appended to that field; if $|F(g)| > 1$, the bigram is ambiguous and is discarded; and if $F(g) = \emptyset$, it remains unmatched. Only bigrams are used (not longer n-grams), and no priority order between fields is applied. Every field of S_k is a list to which

all matches are appended, so several values (for example, several laboratory thresholds or doses) are all kept, and none overwrites another. Tokens that match no dictionary entry form the residual text ρ_k, which is not included in S_k; the structured prompt is therefore a lossy view of the raw prompt. Matching is a dictionary lookup per token or bigram, so its cost grows linearly with prompt length. Algorithm 1 formalizes the procedure.

Algorithm 1 Dictionary- and Ontology-Guided Structured Prompt Generation (SPE-ECG)

Require: Preprocessed prompt p_k; dictionaries D_f for $f \in \mathcal{F}$ derived from $\mathcal{O}$
Ensure: Structured JSON object S_k; residual text ρ_k
Initialize $S_k[f] \leftarrow [\,]$ for every $f \in \mathcal{F}$
Tokenize p_k and assign POS tags
for each noun w in p_k **do**
  if $w \in D_{drug}$ **then**
    Append w to S_k[drug]
  end if
end for
for each verb w in p_k **do**
  Append w to S_k[task]
end for
Form the bigrams B of p_k (bag-of-words step)
for each bigram $g \in B$ **do**
  $F(g) \leftarrow \{f : g \in D_f\}$
  if $|F(g)| = 1$ **then**
    Append g to $S_k[f]$ for the single $f \in F(g)$
  else if $|F(g)| > 1$ **then**
    Discard g (ambiguous)
  end if
end for
$\rho_k \leftarrow$ tokens not matched by any rule
return S_k, ρ_k

3.4.4. Automation and Design Considerations

The SPE-ECG pipeline combines automated linguistic processing with curated domain knowledge. Tokenization, POS tagging, and bigram generation are fully automated. Domain dictionaries are manually curated to ensure semantic precision but are accessed programmatically during extraction.

This hybrid design balances transparency and scalability. While dictionary construction requires initial domain expertise, subsequent structuring operates automatically across the entire corpus.

3.4.5. Methodological Limitations

Despite its effectiveness, the proposed structuring approach presents several limitations:

- **Field Ambiguity:** A bigram that belongs to more than one semantic category is discarded, so such content is lost, and no priority order is applied.
- **Bigram-only Matching:** Expressions longer than two words are not matched as units.
- **Context Sensitivity:** Static dictionary matching may fail to capture contextual meaning shifts.
- **Schema Flatness:** The current JSON schema represents single-level fields and does not model hierarchical dependencies.
- **Domain Transfer Overhead:** Applying SPE-ECG to new domains requires dictionary adaptation and ontology extension.

- **Dictionary Coverage and Quality:** Content that matches no dictionary entry is not passed on to the structured prompt, and the dictionaries here were compiled from the evaluation corpus itself, which may overstate coverage on unseen documents.
- **Information Loss:** The structured prompt is a lossy view of the raw prompt; instructions outside the schema are dropped (see the worked example in Section Worked Example).

Future work may incorporate contextual embedding alignment or graph-based schema representations to address these constraints.

3.5. JSON Structuring Through NLP Process

Figure 3 illustrates the operational realization of the transformation function $\Phi(p_k, O)$ introduced in the previous subsection. The NLP pipeline serves as the computational mechanism that maps unstructured textual prompts to structured JSON representations by systematically identifying semantically relevant components.

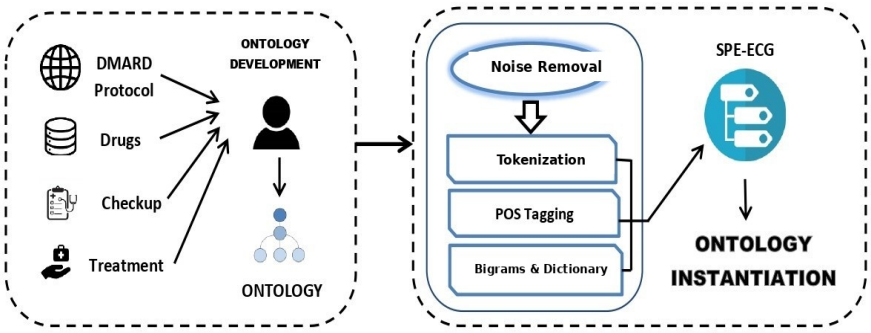


Figure 3. Prompt Structuring Using SPE-ECG. Example instantiated with RA treatment guidelines, but adaptable to other knowledge-intensive domains such as finance, law, and engineering.

The structuring process proceeds through four sequential stages:

1. **Tokenization and Segmentation:** The prompt is decomposed into lexical units $\{w_1, w_2, \ldots, w_m\}$, preserving numerical expressions and domain-specific terminology.
2. **Part-of-Speech Tagging:** Grammatical roles are assigned to tokens to distinguish action verbs (e.g., "monitor," "prescribe") from entities or parameter descriptors.
3. **Bigram Generation:** Two-word sequences are formed with a bag-of-words step to capture compound procedural expressions (e.g., "every 2," "2 weeks").
4. **Ontology-Guided Field Assignment:** Nouns are matched against the drug dictionary, verbs fill the task field, and bigrams are matched against the ontology-derived dictionaries of the remaining fields to populate the structured JSON (Algorithm 1).

Formally, the pipeline implements:

$$S_k = \Phi(p_k, O) = \mathcal{A}(\mathcal{N}(\mathcal{T}(p_k))) \tag{21}$$

where:

- $\mathcal{T}(\cdot)$ denotes tokenization;
- $\mathcal{N}(\cdot)$ denotes bigram expansion and linguistic normalization;
- $\mathcal{A}(\cdot)$ denotes ontology-based assignment.

This layered decomposition ensures that semantic roles are explicitly separated prior to model inference. As shown in Figure 3, domain entities, procedural constraints, and scheduling terms are individually extracted and mapped to predefined schema fields, reducing implicit inference requirements for the LLM.

Unlike conventional free-text prompting, where semantic interpretation is delegated entirely to the model, the proposed structuring process externalizes domain knowledge into an interpretable intermediate representation. This explicit structuring is intended to improve reproducibility and to reduce ambiguity propagation into downstream code generation.

Worked Example

Figure 1 shows the structured prompt produced for the methotrexate protocol, reproduced below. The values are the text segments assigned to each field. The superscript in the laboratory units was lost when the page was scraped and appears as 10^9/L in the extracted text; it is shown here as 10^9/L.

```
{
  ``task'': ``Write the protocol in python'',
  ``language'': ``Python'',
  ``diagnosis'': ``Baseline Then every 2 weeks until dose of
                methotrexate and monitoring has been stable
                for 6 weeks Thereafter every 4 weeks'',
  ``drug'': ``Methotrexate'',
  ``Checkup'': ``WBC <3.5 x 10^9/L Neutrophils <2.0 x 10^9/L
              Platelets <150 x 10^9/L'',
  ``monitoring_protocol'': ``Discuss with specialist team immediately''
}
```

Two observations from this example illustrate the limits of the structuring. First, the dosing and monitoring schedule appears under the key `diagnosis`, so the key names do not always describe their content. Second, the phrase "Complete blood count (CBC)" that appears in the raw prompt of Figure 1 is not present in the structured prompt, which is an instance of the information loss noted in Section 3.4.3.

3.6. Code Generation Protocol

Following structured prompt generation, each JSON object S_k is supplied to the selected language models to generate code:

$$C_{j,k}^{struct} = M_j(S_k) \tag{22}$$

where S_k is the structured prompt, $M_j \in \{M_{GPT3.5}, M_{LLaMA2}, M_{Gemma}\}$, and $C_{j,k}^{struct}$ is the generated code. The raw condition uses $C_{j,k}^{raw} = M_j(p_k)$. Inference settings are identical across conditions (Table 3); no few-shot examples are used. The structured JSON is submitted without an added instruction, whereas the raw prompt includes one (Section 3.1.2).

Unlike unstructured prompts, which require the model to infer semantic roles from free text, the structured JSON input separates components into predefined fields (`task`, `language`, `drug`, `diagnosis`, `Checkup`, and `monitoring_protocol`). The structure is intended to make the required elements explicit to the model; whether it does so in practice is evaluated empirically in Section 4, and the worked example in Section Worked Example shows the form of the input.

4. Results

This section reports the comparison of raw and structured prompts across GPT-3.5, LLaMA 2, and Gemma on the RA case study ($M = 43$ prompts). BLEU is bounded in $[0, 1]$ and measures lexical overlap with the raw-prompt code produced by the other two models; the results should be read as evidence about lexical agreement between models only, and not about correctness.

4.1. Lexical Alignment: Raw vs. Structured Prompts

Table 5 and Figure 4 report mean BLEU for each model and condition, and Figure 5 shows the per-prompt scores. Mean BLEU was higher under structured prompting for all three models. The paired differences were statistically significant for GPT-3.5 and LLaMA 2 (Holm-adjusted $p = 0.003$ for both; the structured prompt scored higher on 30 of 43 prompts for each model and lower on 13). For Gemma, the mean difference was +0.126 (95% CI 0.020 to 0.233), and the structured prompt scored higher on 26 of 43 prompts and lower on 17, but the Wilcoxon test did not reach significance after correction ($p = 0.0505$), so the Gemma result is a trend. Rank-biserial effect sizes were 0.56 (GPT-3.5), 0.55 (LLaMA 2), and 0.34 (Gemma). The Gemma mean gain is also affected by a few prompts on which the structured score fell sharply (for example, prompt 2 fell from 0.824 to 0.016).

Table 5. Mean BLEU (0–1 scale; SD in parentheses) under raw and structured prompting over $M = 43$ prompts per model. Relative change is computed from the unrounded means. The mean paired difference (structured minus raw) is given with a 95% percentile-bootstrap confidence interval (10,000 resamples); *p*-values are from two-sided Wilcoxon signed-rank tests with Holm correction across the three models.

Model	Raw, Mean (SD)	Structured, Mean (SD)	Rel. Change (%)	Mean Difference [95% CI]	*p* (Holm)
GPT-3.5	0.385 (0.276)	0.598 (0.247)	+55.1	0.212 [0.099, 0.324]	0.003
LLaMA 2	0.533 (0.268)	0.706 (0.210)	+32.5	0.173 [0.075, 0.270]	0.003
Gemma	0.632 (0.284)	0.758 (0.201)	+19.9	0.126 [0.020, 0.233]	0.0505

The dispersion of per-prompt scores was lower under structured prompting for all three models (standard deviation 0.276 to 0.247 for GPT-3.5, 0.268 to 0.210 for LLaMA 2, and 0.284 to 0.201 for Gemma; Table 6). This is a descriptive observation only, and no inferential claim about variance is made. Raw-prompt BLEU is highest for Gemma and lowest for GPT-3.5. BLEU depends on output length and style (through the brevity penalty and n-gram precision), so differences in absolute BLEU between models should not be read as differences in code quality; output length and brevity penalty were not analyzed, so the reason for this ordering is not known.

Because the references are the raw-prompt outputs of the other two models, a higher BLEU under structured prompting means that the structured-prompt code resembles what the other models write from raw prompts more closely. It does not mean that the code is more correct. Likewise, the raw condition scores raw outputs against other raw outputs, whereas the structured condition scores structured outputs against raw outputs, so the two sets of scores are not strictly like-for-like.

Table 6. Distribution of per-prompt BLEU scores by model and condition: median, interquartile range (IQR), and range.

	Raw			Structured		
Model	Median	IQR	Range	Median	IQR	Range
GPT-3.5	0.420	0.505	0.045–0.879	0.613	0.379	0.076–0.980
LLaMA 2	0.551	0.357	0.048–0.987	0.748	0.351	0.228–0.999
Gemma	0.737	0.301	0.052–0.987	0.811	0.233	0.016–0.993

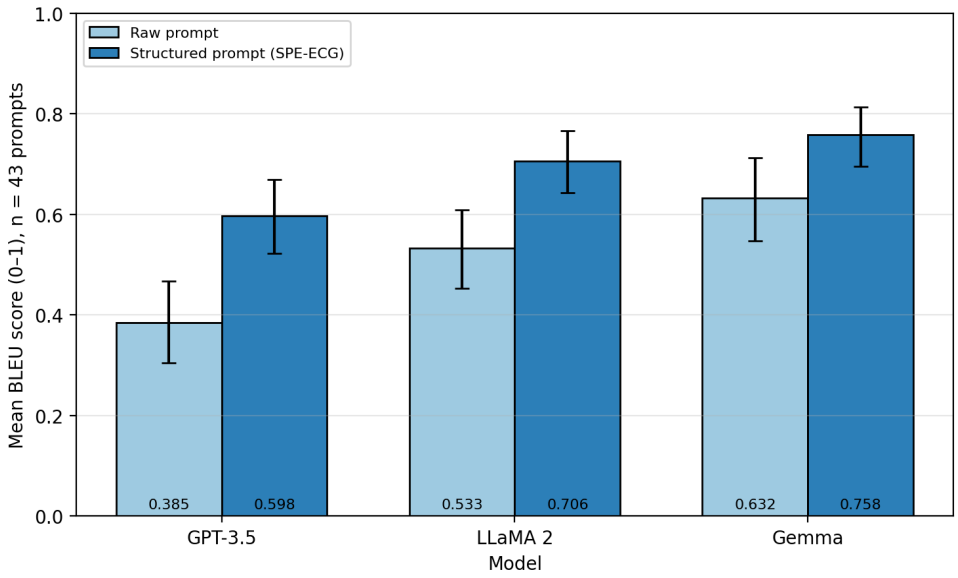


Figure 4. Mean BLEU for raw and structured prompting for each model ($M = 43$ prompts), with 95% percentile-bootstrap confidence intervals.

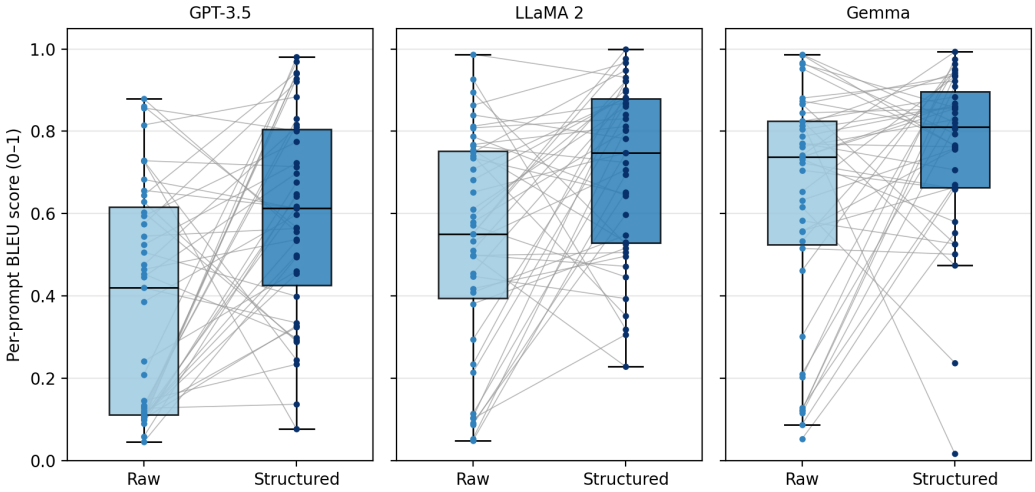


Figure 5. Per-prompt BLEU ($n = 43$) for raw and structured prompts for each model. Boxes show the median and interquartile range, points are individual prompts, and gray lines connect the paired scores of each prompt. All values lie in $[0, 1]$.

4.2. Qualitative Analysis and Failure Patterns

All generated outputs of the three models were read by the authors (no clinician reviewed them), and this reading indicated that, under structured prompting, dosage schedules, monitoring intervals, compound expressions (e.g., "every 2 weeks until stable for 6 weeks"), and multi-word entities (e.g., "liver function test") were more often reproduced than under raw prompting, where multi-stage instructions were more often simplified. These observations are qualitative and were not quantified. Failure patterns persisted under structured prompting:

- **Incomplete logical implementation:** Generated code sometimes included the entities and parameters but omitted conditional branching or escalation logic; for example, dosage intervals were preserved while safety checks were occasionally excluded. Lexical alignment therefore does not guarantee procedural completeness.

- **Schema field misinterpretation:** Multi-word keys such as `monitoring_protocol` were occasionally treated as narrative text, which suggests that shorter, atomic field names may reduce ambiguity.
- **Temporal dependency loss:** Sequential or conditional dependencies (e.g., "increase dose after 3 weeks if no improvement") are difficult to encode in a flat JSON schema, and multi-stage dependencies were sometimes simplified.

The frequency of these patterns, including the omission of safety checks, was not quantified in this study.

5. Discussion

5.1. Interpretation of the Findings

In this case study, mean BLEU was higher under SPE-ECG structured prompting than under raw prompting for all three models. This indicates higher lexical agreement with the raw-prompt code of the other two models. It does not show that the generated code is more correct, more reliable, or clinically safer, because BLEU does not execute code, and no functional evaluation was performed. The qualitative observation that conditional logic and safety checks were sometimes omitted even under structured prompting (Section 4) shows that lexical overlap and procedural completeness can diverge.

The comparison of raw prompts with the complete pipeline changes several components at once (preprocessing, dictionary-based selection of content, field labels, and JSON formatting). The present results therefore cannot be attributed to ontology guidance specifically; an ablation study is needed to address this and is listed as future work. The gain was statistically significant for GPT-3.5 and LLaMA 2 but not for Gemma after correction for multiple comparisons, and Gemma had the highest raw-prompt baseline. Input-level structuring is nonetheless a lightweight option that does not modify model parameters, and it may complement scaling or fine-tuning.

Two further points limit the interpretation. The references are other models' raw-prompt outputs, so BLEU rewards resemblance to what the other models write from raw prompts and not correctness. The raw and structured prompts also differed in wording: the raw prompt carried an explicit instruction and the table headers, whereas the structured prompt did not carry a separate instruction. Model differences should also be interpreted cautiously. GPT-3.5, LLaMA 2, and Gemma are all general-purpose models. The larger relative gain for GPT-3.5 coincides with its lower raw-prompt BLEU, and a ratio of means is sensitive to the size of the baseline. The finding that raw-prompt BLEU is higher for Gemma than for GPT-3.5 has not been analyzed and may reflect differences in output length and style rather than code quality (Section 4).

5.2. Positioning Within the Code Generation Landscape

Existing work in code generation primarily follows three directions: (i) large-scale transformer architectures (e.g., Codex, AlphaCode), (ii) instruction-tuned models (e.g., FLAN-T5, T0++), and (iii) domain-specialized pretraining (e.g., BioGPT, MedPaLM, ClinicalBERT).

SPE-ECG is intended to complement these approaches by operating at the input representation layer rather than the model layer. Unlike fine-tuning or pretraining strategies, which require substantial computational resources and domain-specific datasets, SPE-ECG can be implemented as a preprocessing pipeline, which makes it lightweight; it was tested with three models, so its model-agnostic character is supported only to that extent. Table 7 summarizes the positioning of SPE-ECG relative to existing approaches.

Table 7. Comparison of existing approaches with SPE-ECG.

Approach	Strengths	Limitations Compared to SPE-ECG
Rule-based Systems	Transparent, deterministic outputs	Limited scalability; weak handling of unstructured language
Transformer Models	Strong general-purpose generation	Sensitive to ambiguous prompts; limited domain precision
Instruction-Tuned Models	Improved task-following	Still rely on unstructured inputs
Domain-Specific LLMs	Strong terminology understanding	High resource cost; limited code-generation focus
SPE-ECG (Proposed)	Lightweight; no fine-tuning; higher BLEU in the RA case study	Rule/dictionary dependency; flat schema; single-domain evaluation; functional correctness not established

5.3. Applicability Beyond the RA Case Study

SPE-ECG was evaluated only on RA/DMARD monitoring protocols. Applying it to another domain requires new dictionaries and a new ontology (Section 3.4.2), the effort for which was not measured in this study. Potential target areas, such as financial compliance rules, legal obligations and deadlines, or engineering process constraints, are hypothetical use cases and not results of this study; whether the approach transfers, and at what cost, is untested.

Recommended workflow for a new domain. We suggest the following procedure, which follows the one used here but adds a coverage check that was not performed in this study: (1) choose the schema fields from the decisions the downstream code needs (entities, quantities, schedules, conditions, actions); (2) write an inclusion rule and examples for each field, as in Table 4; (3) list candidate terms from a sample of the source texts and merge variants; (4) check coverage on a small held-out sample of texts, i.e., the fraction of tokens matched by the dictionaries, and add missing terms; (5) review the lists with a domain expert; and (6) freeze and publish the dictionaries with the extraction code.

5.4. Clinical Safety Considerations

The generated code implements drug-monitoring logic (e.g., methotrexate dosing and monitoring rules). The protocols come from a single guidance article published in 2008, which may not reflect current clinical practice, and the dictionaries and the generated code were not clinically validated (the protocols themselves are used in clinical practice by the doctors who suggested them as study material). None of the code produced in this study should be used for patient care. The qualitative finding that safety checks or escalation logic were sometimes omitted even under structured prompting is a safety risk in any clinical deployment; its frequency was not quantified in this study. Any clinical use would require expert review of the generated code and prospective validation.

5.5. Objectives Achieved and Practical Utility

The study had three objectives. (i) *Design a transparent pipeline* that turns unstructured protocol text into a structured prompt: achieved, and the extraction code and dictionaries are released. (ii) *Evaluate whether structured prompts change the code produced by different LLMs*: achieved within the limits stated in Section 5.7; mean BLEU agreement was higher for all three models and significantly so for GPT-3.5 and LLaMA 2. (iii) *Establish that the generated code is correct and clinically safe*: not achieved, because no functional or clinical evaluation was performed.

With regard to practical utility, the pipeline needs no fine-tuning, no specialized hardware, and no paid model; it was run with a free web model and two open models on a local machine. Because every step is rule-based and every dictionary entry can be read, the structured prompt can be audited, which is relevant in clinical settings. Possible uses are as a preparation step in tools that turn clinical monitoring protocols into checklists or reminder scripts, with human review of the output. The results do not show that the approach is ready for deployment. That would require clinical validation, execution-based tests, additional protocols and domains, and comparison with state-of-the-art baselines.

5.6. Domain-Specific Observations

Three observations are specific to monitoring protocols. First, the schema separates the elements that matter in a monitoring protocol: The drug, the laboratory tests with their thresholds (`Checkup`), the schedule, and the action to take (`monitoring_protocol`). Second, the relative gain was largest for the model with the lowest raw-prompt score (GPT-3.5, raw 0.385) and smallest for the model with the highest (Gemma, raw 0.632), which is consistent with structuring helping most where raw prompts work worst; this pattern was observed but not tested. Third, the failure patterns are typical of protocols: Conditional and staged schedules (for example, "every 2 weeks until stable for 6 weeks, then every 4 weeks") cannot be represented in a flat schema, and in the example of Section Worked Example the schedule is stored under the key `diagnosis`, so the key names do not describe their content. A domain-informed redesign of the schema, with separate fields for frequency, duration, threshold, and action, and with support for conditional stages, is the most direct improvement suggested by these observations.

5.7. Limitations and Threats to Validity

- **Lexical-overlap metric.** BLEU measures n-gram overlap and not functional correctness; claims about reliability, procedural completeness, or safety are not supported by BLEU alone. No execution-based, code-aware, or expert evaluation was performed.
- **No independent reference.** Each model's code was scored against the raw-prompt code of the other two models. BLEU therefore measures agreement between models, and the reference set is itself raw-prompt output, which is the baseline the method aims to improve. High agreement with such references is not evidence of correct code, and the raw and structured scores are not strictly like-for-like.
- **Prompt wording differs between conditions.** The raw prompt contained the instruction "write the protocol in python language" and the table headers; the structured prompt was the JSON alone. Part of any difference may come from this wording and not from the structure.
- **Confounded comparison.** The raw-versus-full-pipeline comparison does not isolate ontology guidance from formatting, extraction, or content selection; no ablation was performed.
- **Information loss.** Content that matches no dictionary entry, and bigrams that match more than one field, are not passed to the model, so the two prompt sets are not informationally equivalent, and coverage was not measured.
- **Dictionary-based "ontology".** The ontology is a small, hand-built entity–attribute schema compiled by the authors from the evaluation corpus in about one week and reviewed by no independent expert; its relations are not used, and it is not aligned to standard terminologies.
- **Single source and small sample.** All 43 protocols come from one web page (a 2008 New Zealand guidance article), the dictionaries were compiled from that page, and there is one prompt per protocol.

- **Models and sampling.** GPT-3.5 was used through the free ChatGPT interface in June–July 2024 and LLaMA 2 (13B) and Gemma (7B) through Ollama, all with default settings that were not controlled. All 43 prompts were submitted in a single chat per model, so earlier prompts may have influenced later outputs, and one generation was produced per prompt. These are older, general-purpose models; current frontier and code-specialised models were not included.
- **Statistical power.** With 43 prompts per model, the Gemma difference did not reach significance after correction; results for individual models should be interpreted with this in mind.
- **Rule-based fragility.** Paraphrases can bypass dictionary matching, and the flat schema cannot represent nested or sequential dependencies.
- **No state-of-the-art baselines.** SPE-ECG was compared only with raw prompting of the same three models. Its performance relative to other prompting methods, fine-tuned models, and newer or code-specialized models is unknown.

6. Conclusions

This study introduced SPE-ECG, a rule-based framework that converts unstructured domain text into a structured JSON representation using NLP preprocessing, curated dictionaries, and a lightweight domain ontology, and evaluated it in a single case study (RA/DMARD monitoring protocols) with three LLMs. Mean BLEU against the raw-prompt code of the other two models was higher under structured prompting for all three models, with relative gains of 55.1% (GPT-3.5), 32.5% (LLaMA 2), and 19.9% (Gemma); the gains were statistically significant for GPT-3.5 and LLaMA 2 but not, after correction for multiple comparisons, for Gemma (Section 4).

These results indicate higher lexical agreement of structured-prompt code with other models' raw-prompt code. They do not demonstrate functional correctness, clinical safety, reliability, or generalization to other domains, and the current design cannot separate the effect of ontology guidance from that of reformatting. Ablation studies, execution-based and expert evaluation, and tests on additional domains and current models are needed before stronger claims can be made. Within these limits, input-level structuring is a lightweight, fine-tuning-free approach that merits further investigation.

7. Future Work

Several avenues exist for extending the SPE-ECG framework:

- **Ablation and Baselines:** Evaluate ablation and baseline conditions (JSON without dictionaries, extraction without ontology, explicit natural-language prompts, LLM-based extraction, and few-shot prompting) to isolate the contribution of ontology guidance.
- **Independent References:** Write independent, clinician-validated reference implementations and re-score all conditions against them.
- **Functional and Clinical Evaluation:** Complement BLEU with CodeBLEU, execution-based tests (pass@k), and expert rating of clinical constraints, and quantify the omission of safety checks.
- **Standards-Based Ontology:** Replace the keyword dictionaries by a formal ontology (e.g., OWL/RDF) aligned to RxNorm, ATC, SNOMED CT, or UMLS, and use the relations R in extraction.
- **Hierarchical Schema Design:** Extend the flat JSON structure to hierarchical or graph-based representations (e.g., JSON-LD, RDF) to better model sequential and nested procedural dependencies.

- **Hybrid Extraction Methods:** Integrate rule-based structuring with learned schema induction or transformer-based extraction models to improve robustness against linguistic variation.
- **Domain Expansion and Current Models:** Apply the methodology to other domains (e.g., finance, law, engineering) and evaluate current frontier and code-specialized models (e.g., CodeGemma, DeepSeek Coder) and fine-tuned baselines.
- **State-of-the-Art Baselines:** Compare SPE-ECG with few-shot prompting, structured chain-of-thought prompting, LLM-based extraction into the same schema, and current frontier and code-specialized models, and evaluate with execution-based tests and expert review.

These extensions will clarify the role of structured prompt engineering in domain-specific code synthesis.

Author Contributions: Conceptualization, T.M., M.T.A. and G.M.; methodology, T.M.; software, T.M.; validation, T.M., M.T.A. and G.M.; formal analysis, T.M.; investigation, T.M.; resources, T.M.; data curation, T.M.; writing—original draft preparation, T.M.; writing—review and editing, M.T.A. and G.M.; visualization, T.M.; supervision, M.T.A.; project administration, M.T.A., A.S.A.-S., I.D.l.T.D., L.J.H.R. and P.H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by Fundación Universidad Europea del Atlántico, C/ Isabel Torres 21, 39011 Santander, Spain (Tax Identification Number (CIF): ESG39764972). research received no external funding.

Data Availability Statement: The datasets and code generated and analyzed during the current study are available in the repository at https://github.com/Talha331498/Dataset.git, (accessed on 1 May 2024). The repository contains the raw prompts, the structured (JSON) prompts, the code generated by the three models, the per-prompt BLEU scores, and the extraction code and dictionaries. The source protocols are from the publicly available bpac^{nz} article at https://bpac.org.nz/BPJ/2008/October/dmards.aspx, (accessed on 1 May 2024). The repository includes the dictionary files and the inclusion rules of Table 4 so that the dictionary construction can be repeated or adapted.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT-3.5 (OpenAI), Claude Sonnet 5.5 (Anthropic), and Grammarly 1.2.300 for Windows for language editing, grammar correction, sentence restructuring, and improving the clarity, readability, and overall linguistic quality of the manuscript. These tools were used exclusively to enhance the presentation of the text without altering the scientific content, methodology, results, or conclusions. The authors have carefully reviewed, verified, and edited all AI-assisted outputs and take full responsibility for the content of this publication.

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

References

1. Wan, Y.; Bi, Z.; He, Y.; Zhang, J.; Zhang, H.; Sui, Y.; Xu, G.; Jin, H.; Yu, P. Deep learning for code intelligence: Survey, benchmark and toolkit. *ACM Comput. Surv.* **2024**, *56*, 309. [CrossRef]
2. Zhang, S.; Chen, Z.; Shen, Y.; Ding, M.; Tenenbaum, J.B.; Gan, C. Planning with large language models for code generation. *arXiv* **2023**, arXiv:2303.05510.
3. Nazi, Z.A.; Peng, W. Large language models in healthcare and the medical domain. *Informatics* **2024**, *11*, 57. [CrossRef]
4. Alraba'nah, Y.; Sleit, A.; Qaddara, I.; Hiari, M. Code generation by large language models: A comparative analysis of ChatGPT, Claude, and DeepSeek. *Int. J. Electr. Electron Eng. Telecommun.* **2026**, *15*, 19–28. [CrossRef]
5. Kehkashan, T.; Abdullah, M.; Al-Shamayleh, A.S.; Ivković, N.; Ismail, N.A.; Ahmad, S.S.S.; Rehman, A.; Akhunzada, A. From benchmarks to deployment: A comprehensive review of agentic AI evaluation. *Artif. Intell. Rev.* **2026**, *59*, 167. [CrossRef]
6. Jiang, J.; Wang, F.; Shen, J.; Kim, S.; Kim, S. A survey on large language models for code generation. *arXiv* **2024**, arXiv:2406.00515.
7. Sanh, V.; Webson, A.; Raffel, C.; Bach, S.H.; Sutawika, L.; Alyafeai, Z.; Chaffin, A.; Stiegler, A.; Scao, T.L.; Raja, A.; et al. Multitask prompted training enables zero-shot task generalization. *arXiv* **2021**. [CrossRef]

8. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef] [PubMed]
9. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
10. Tam, Z.R.; Wu, C.K.; Tsai, Y.L.; Lin, C.Y.; Lee, H.Y.; Chen, Y.N. Let me speak freely? A study on the impact of format restrictions on performance of large language models. *arXiv* **2024**, arXiv:2408.02442.
11. Radu, A.F.; Bungau, S.G. Management of rheumatoid arthritis: An overview. *Cells* **2021**, *10*, 2857. [CrossRef] [PubMed]
12. Coskun, B.N.; Yagiz, B.; Ocakoglu, G.; Dalkilic, E.; Pehlivan, Y. Assessing the accuracy and completeness of artificial intelligence language models in providing information on methotrexate use. *Rheumatol. Int.* **2024**, *44*, 509–515. [CrossRef] [PubMed]
13. Gulwani, S.; Polozov, O.; Singh, R. Program Synthesis. *Found. Trends Program. Lang.* **2017**, *4*, 1–119. [CrossRef]
14. Liu, P.; Gao, W.; Dong, W.; Ai, L.; Gong, Z.; Huang, S.; Li, Z.; Hoque, E.; Hirschberg, J.; Zhang, Y. A survey on open Information Extraction from rule-based model to large language model. *arXiv* **2022**, arXiv:2208.08690.
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
16. Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H.P.d.O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. Evaluating Large Language Models Trained on Code. *arXiv* **2021**, arXiv:2107.03374.
17. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv* **2023**, arXiv:2307.09288.
18. Alsentzer, E.; Murphy, J.; Boag, W.; Weng, W.H.; Jindi, D.; Naumann, T.; McDermott, M. Publicly Available Clinical BERT Embeddings. In Proceedings of the 2nd Clinical Natural Language Processing Workshop, Minneapolis, MN, USA, 7 June 2019; pp. 72–78.
19. Li, J.; Li, G.; Li, Y.; Jin, Z. Structured Chain-of-Thought Prompting for Code Generation. *arXiv* **2023**, arXiv:2305.06599.
20. Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 3580–3599. [CrossRef]
21. Gruber, T.R. A Translation Approach to Portable Ontology Specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
22. Bodenreider, O. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Nucleic Acids Res.* **2004**, *32*, D267–D270. [CrossRef] [PubMed]
23. Boxwala, A.A.; Peleg, M.; Tu, S.; Ogunyemi, O.; Zeng, Q.T.; Wang, D.; Patel, V.L.; Greenes, R.A.; Shortliffe, E.H. GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines. *J. Biomed. Inform.* **2004**, *37*, 147–161. [CrossRef] [PubMed]
24. Fox, J.; Johns, N.; Rahmanzadeh, A. Disseminating Medical Knowledge: The PROforma Approach. *Artif. Intell. Med.* **1998**, *14*, 157–182. [CrossRef] [PubMed]
25. Hripcsak, G.; Ludemann, P.; Pryor, T.A.; Wigertz, O.B.; Clayton, P.D. Rationale for the Arden Syntax. *Comput. Biomed. Res.* **1994**, *27*, 291–324. [CrossRef] [PubMed]
26. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
27. Ren, S.; Guo, D.; Lu, S.; Zhou, L.; Liu, S.; Tang, D.; Sundaresan, N.; Zhou, M.; Blanco, A.; Ma, S. CodeBLEU: A Method for Automatic Evaluation of Code Synthesis. *arXiv* **2020**, arXiv:2009.10297.
28. Liu, J.; Xia, C.S.; Wang, Y.; Zhang, L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 21558–21572. [CrossRef]
29. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training Language Models to Follow Instructions with Human Feedback. *Proc. Adv. Neural Inf. Process. Syst.* **2022**, *35*, 27730–27744. [CrossRef]
30. Tipirneni, S.; Zhu, M.; Reddy, C.K. Structcoder: Structure-aware transformer for code generation. *ACM Trans. Knowl. Discov. From Data* **2024**, *18*, 70. [CrossRef]
31. Maharjan, J.; Garikipati, A.; Singh, N.P.; Cyrus, L.; Sharma, M.; Ciobanu, M.; Barnes, G.; Thapa, R.; Mao, Q.; Das, R. OpenMedLM: Prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models. *Sci. Rep.* **2024**, *14*, 14156. [CrossRef] [PubMed]
32. Gemma Team. Gemma: Open Models Based on Gemini Research and Technology. *arXiv* **2024**, arXiv:2403.08295.

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.