# Politecnico di Torino

Master's Degree in Computer Engineering


Politecnico
di Torino

**Master's Degree Thesis**

## Specializing Large Language Models for Biofabrication Protocol Generation through Domain-Specific Fine-Tuning

**Supervisors**

Prof. Roberta Bardini

Prof. Stefano Di Carlo

Prof. Alessandro Savino

M.Sc. Riccardo Smeriglio

**Candidate**

Selen Karakaş

March 2026

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisors, Prof. Stefano Di Carlo, Prof. Alessandro Savino, Prof. Roberta Bardini, and Riccardo Smeriglio for their guidance and support throughout this research. I am especially grateful to Prof. Roberta Bardini and Riccardo Smeriglio for their continuous support, valuable feedback, and availability for discussion during this work. Their guidance played a key role in shaping this thesis and in helping me navigate the technical and research challenges of the project. I would also like to thank the members of the SMILIES research group for providing such a stimulating and supportive research environment.

Beyond the academic environment, this journey has also been deeply personal. *Öncelikle, yalnızca bu süreçte değil hayatımın her anında bana inanan, her koşulda yanımda olan ve sevgilerini her zaman hissettiğim anneme ve babama teşekkür ederim. Bana verdiğiniz destek, sabır, belki de en çok sabır, ve güven olmasaydı bugün burada olamazdım. Evimizin en minik üyesi Elma, zor günlerde bile yüzümü güldürmeyi hiç ihmal etmediğin için sana da ayrıca teşekkür ederim. Hepiniz iyi ki varsınız.*

*Torino'nun bir şehirden çok ev gibi hissettirmesini sağlayan ailem, Ecem, Özge ve Koray; sadece birlikte paylaştığımız anılar ve kahkahalar için değil her koşulda yanımda olduğunuz için size ne kadar teşekkür etsem az, iyi ki varsınız. Bayram, Meriç ve Oğuzhan; dostluğunuz, desteğiniz ve hayatıma kattığınız tüm güzel anlar için teşekkür ederim, iyi ki varsınız.* I would also like to thank everyone I had the chance to meet here in Torino and who, in one way or another, became part of my life during this time. You made this journey far more enjoyable and meaningful.

Looking back, this thesis represents more than an academic milestone. It reflects a journey of learning, growth, challenges, and unforgettable moments. The people I met and the friendships I built during this time have shaped not only this work but also who I am today.

To everyone who has been part of this, in one way or another, thank you. And for all the memories that came with this journey, I will always be grateful.

# ABSTRACT

Biofabrication is the process of creating biological structures, such as tissues or scaffolds, using living cells and biomaterials. In the laboratory, these processes are described in protocols that provide step-by-step instructions for preparing materials, handling cells, and setting fabrication parameters. These protocols are critical for reproducibility. However, they are usually written in free-form natural language, making them difficult to standardize, compare, or use within computational systems. While formal protocol languages and ontology-based frameworks have been proposed, their adoption remains limited due to the manual effort required to convert existing procedural text into structured formats. Machine-readable protocol formats refer to representations that computational models, digital tools, or automated laboratory systems can directly interpret. As experiments become more complex, the lack of structured and machine-readable protocol formats limits automation and scalability. Manually converting existing textual protocols into structured formats is time-consuming and does not scale to large collections of documents.

This thesis addresses this challenge by developing an end-to-end framework for large-scale data collection, normalization, and model adaptation. A large corpus of biofabrication protocols is collected from heterogeneous public sources, including scientific articles indexed in databases such as PubMed and open online protocol repositories such as Protocol.io. These documents are processed through a normalization pipeline that restructures unstructured procedural text into explicitly ordered steps. The pipeline includes automated extraction of action verbs and experimental parameters, step segmentation, terminology standardization, and structured formatting. To support large-scale processing, the Gemini Application Programming Interface (API) is used to assist in semantic parsing and action normalization across heterogeneous protocol sources. The processed dataset is used to fine-tune a transformer-based large language model Large Language Model Meta AI (LLaMA)-3.1-8B-Instruct using Quantized Low-Rank Adaptation (QLoRA). The objective is to adapt the model to the biofabrication domain to generate structured, consistent, and both human- and machine-readable biofabrication protocols. The fine-tuning process focuses on teaching the model to reproduce explicit step structures, standardized terminology, and accurate experimental parameters (e.g., temperature) within procedural descriptions. Experimental results show that combining domain-specific preprocessing with model fine-tuning improves the clarity, consistency, and structural quality of generated protocols. The proposed framework contributes toward scalable digital documentation of laboratory procedures and represents a step toward integrating intelligent language models with automated and semi-automated laboratory systems.

# Contents

# ACRONYMS

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **NLP** | Natural Language Processing |
| **LLM** | Large Language Model |
| **LLMs** | Large Language Models |
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **GPT** | Generative Pre-trained Transformer |
| **LLaMA** | Large Language Model Meta AI |
| **QLoRA** | Quantized Low-Rank Adaptation |
| **API** | Application Programming Interface |
| **JSON** | JavaScript Object Notation |
| **JSONL** | JSON Lines |
| **RDF** | Resource Description Framework |
| **PAML** | Protocol Activity Markup Language |
| **UML** | Unified Modeling Language |
| **SBOL** | Synthetic Biology Open Language |
| **PCR** | Polymerase Chain Reaction |
| **BLEU** | Bilingual Evaluation Understudy |
| **ROUGE** | Recall-Oriented Understudy for Gisting Evaluation |
| **BioLP-bench** | Biological Laboratory Protocol Benchmark |
| **BioProBench** | Biological Protocol Benchmark |

# Chapter 1

# INTRODUCTION

Biofabrication is a rapidly developing field of experimental science that involves complex laboratory procedures to create tissues, scaffolds, and engineered biological constructs through the controlled combination of cells, biomaterials, and fabrication technologies [1,2]. Reproducibility has become one of the central challenges in modern experimental science, particularly in domains that rely on complex laboratory procedures. Biofabrication is one such domain, focused on the creation of tissues, scaffolds, and engineered biological constructs using cells, biomaterials, and controlled fabrication processes [1, 2]. As biofabrication techniques become more sophisticated, the protocols describing these procedures play an increasingly important role in ensuring that experiments can be understood, repeated, and validated. Despite their importance, these protocols are typically written in unstructured natural language, which makes them difficult to interpret consistently and even harder to reproduce or automate [1, 2]. The field has expanded rapidly over the past decade, with advances in bioprinting, scaffold engineering, and tissue fabrication contributing to increasingly intricate experimental workflows. Foundational assessments of the field have emphasized the rapid growth of new technologies and the need for clearer terminology and methodological consistency [1,2]. As methods evolve, the protocols supporting them have grown longer, more detailed, and increasingly sensitive to procedural accuracy.

Biofabrication relies fundamentally on laboratory protocols and procedural descriptions that guide researchers through material preparation, cell handling, equipment operation, and fabrication steps [1, 2]. These documents serve as the basis for experimental reproducibility, enabling one laboratory to reproduce another's work under comparable conditions [3]. However, protocols written in free-form natural language vary substantially in structure, level of detail, and terminology, often leading to ambiguity [4]. Minor differences in phrasing can result in significant variations in experimental outcomes [3]. Missing temperatures, unclear instructions such as unspecified incubation durations, or inconsistent material descriptions challenge even experienced practitioners and render compu-

tational interpretation nearly impossible [5, 6]. As biofabrication becomes more closely integrated with automation and data-driven methodologies, the limitations of unstructured protocols become increasingly restrictive [4, 7].

To mitigate these challenges, researchers have proposed structured representations of laboratory procedures. One of the earliest and most influential efforts in this direction is BioCoder, a domain-specific programming language designed to express biological workflows in a standardized, computationally interpretable format [8]. BioCoder demonstrated that complex experimental procedures could be formalized using constructs similar to those found in programming languages, thereby enhancing clarity, reproducibility, and the potential for automation. More recent work has examined the need for unified, machine-readable formats, such as open protocol representations intended to support consistent documentation and computational interoperability [4]. Despite their promise, these approaches are not widely adopted because converting the large number of existing natural-language protocols into structured forms requires significant manual effort.

In parallel, rapid advances in Artificial Intelligence (AI) have introduced new opportunities to address these issues, in particular, thanks to Large Language Models (LLMs). LLMs are deep neural network models based on the transformer architecture, trained on large-scale text corpora to capture linguistic patterns and generate coherent natural language [9].By learning contextual representations of language from vast amounts of data, these models have significantly advanced a wide range of natural language processing tasks, including text understanding, generation, and reasoning [10–12]. Recent studies also highlight the growing role of large language models in biomedical research, where they are increasingly used for literature analysis, biological reasoning, and experimental knowledge extraction [13, 14].

Alongside these general-purpose models, a growing number of scientific and biomedical LLMs have been developed to better capture domain-specific terminology and knowledge. Scientific language models are LLMs typically trained or further adapted using specialized scientific corpora, enabling improved performance in technical domains and tasks requiring scientific reasoning.Their ability to interpret and generate complex procedural text suggests a promising direction for addressing challenges in interpreting and generating laboratory protocols [15–17].

Although these developments highlight the potential of language models for scientific applications, their use in biofabrication remains limited [7, 18]. Recent research has begun exploring the use of large language models for protocol generation, experimental planning, and automated laboratory workflows [19, 20]. Existing models are not trained to understand the procedural logic of laboratory workflows, nor are they designed to generate the precise, unambiguous instructions required for reproducible experiments [5, 6]. Furthermore, the absence of large, domain-specific datasets prevents LLMs from learning the structured patterns

common in biofabrication, such as hydrogel formulation, cell seeding strategies, or multi-step bioprinting operations [7]. The gap between unstructured protocol text and the need for structured, reproducible procedures presents a clear and compelling research opportunity.

The work presented in this thesis addresses this opportunity by focusing on the lack of standardized, machine-readable, and reproducible representations for biofabrication protocols. This challenge is practically relevant because biofabrication experiments rely heavily on procedural accuracy, and the ability to formalize protocols is essential for supporting reproducibility, automation, and scalability in experimental design. The core contribution of this thesis is the development of a framework that integrates large-scale protocol collection, a comprehensive normalization pipeline, and the adaptation of transformer-based language models for structured protocol generation. The methodology includes gathering tens of thousands of protocols from scientific repositories, converting them into consistent formats, and adapting a transformer-based LLM through parameter-efficient fine-tuning (e.g., QLoRA) [21] to generate clear and unambiguous procedural steps. The approach is grounded in literature analysis, computational modeling, and empirical evaluation.

This manuscript is organized as follows. Chapter 2 provides background on biofabrication, biological protocols, and transformer-based language models, and reviews related scientific work. Chapter 3 presents the methods used in this research, including dataset collection, normalization pipelines, model design, and training procedures. Chapter 4 reports the results obtained from the fine-tuned models and evaluates their performance across quantitative and qualitative criteria. Chapter 5 discusses the implications of these results, their relevance within the broader scientific context, and the limitations of the current approach. Chapter 6 concludes the thesis by summarizing the contributions and suggesting future directions, particularly in relation to automated laboratory execution and intelligent protocol design.

# Chapter 2

# BACKGROUND AND RELATED WORK

This chapter provides the scientific and technical background necessary to understand the contributions of this thesis. It first introduces the field of biofabrication and the role of experimental protocols in documenting complex laboratory workflows. The chapter then reviews existing approaches for representing biological procedures in structured and machine-readable formats. Finally, it presents recent advances in transformer-based natural language processing and large language models, with a particular focus on their application to scientific text and experimental protocols. Together, these topics provide the conceptual and technological context for the work presented in this thesis.

## 2.1 Biofabrication and Experimental Protocols

Biofabrication has emerged as a rapidly evolving research field that integrates biological systems, biomaterials, and engineering approaches to create functional three-dimensional biological constructs. The term biofabrication broadly refers to the automated generation of biologically functional products using living cells, biomaterials, and bioactive molecules [1, 2]. These constructs are widely studied in areas such as regenerative medicine, drug testing, and tissue engineering where engineered tissues can be used to replace damaged biological structures or serve as in vitro experimental models. As fabrication technologies have advanced, experimental workflows have become increasingly complex, often involving multistep laboratory procedures that combine cell preparation, biomaterial processing, and controlled fabrication processes [1].

The concept of biofabrication has evolved significantly over the past two decades as advances in biomaterials, cell engineering, and additive manufacturing technologies have converged. Early tissue engineering approaches primarily relied on

scaffold-based strategies in which cells were seeded onto pre-fabricated biomaterial structures. However, the emergence of additive manufacturing techniques enabled the direct spatial organization of living cells and biomaterials during fabrication, allowing researchers to design complex biological architectures with greater precision [22, 23]. These developments transformed biofabrication from a largely experimental laboratory technique into a multidisciplinary engineering field combining principles from biology, materials science, mechanical engineering, and computer-aided manufacturing [1, 2].

Biofabrication workflows typically involve multiple interconnected stages that extend beyond the fabrication step itself. A typical pipeline includes the preparation of cells and biomaterials, the formulation of bioinks, the fabrication process itself (for example through bioprinting or microfabrication), and post-fabrication maturation processes such as cell culture, differentiation, and biochemical conditioning [2]. Each of these stages may involve several experimental parameters including reagent concentrations, temperature conditions, incubation times, and mechanical properties of the materials used. As a result, experimental protocols in this domain often contain a high density of procedural information that must be carefully documented to ensure reproducibility.

Beyond tissue engineering, biofabrication technologies have found applications in a wide range of biomedical and research contexts. Engineered tissues are increasingly used as in vitro models for drug screening, disease modeling, and personalized medicine, where patient-derived cells can be used to reproduce specific pathological conditions [23]. Organ-on-chip systems further integrate microfluidic technologies with engineered tissues to simulate physiological environments and improve the predictive power of laboratory models. These developments highlight the importance of reproducible experimental workflows, since even small variations in protocol execution may significantly affect experimental outcomes.

Biofabrication encompasses a range of advanced manufacturing approaches for organizing cells and biomaterials in three-dimensional space. Among the most widely studied methods are extrusion-based, inkjet, and laser-assisted bioprinting, which enable controlled spatial deposition and the fabrication of tissue-like constructs with tailored material and cellular properties [22, 23]. These techniques have made biofabrication an important enabling technology for engineered tissues and advanced in vitro biomedical models.

The terminology, scope, and methodologies of biofabrication have continually evolved, reflecting rapid technological development and the emergence of advanced fabrication strategies such as extrusion-based bioprinting, light-assisted fabrication, and scaffold-free cell assembly [2]. These technologies enable increasingly precise spatial control over biological materials, allowing the creation of complex tissue architectures. Despite this technological progress, the documentation of experimental workflows has not evolved at the same pace, leaving a gap between

advanced fabrication capabilities and standardized procedural representation.

A central characteristic of biofabrication is its strong dependence on procedural accuracy. Experimental workflows typically involve multi-step laboratory operations, including material preparation, cell handling, bioprinting parameters, and post-fabrication culture conditions. These steps are documented through laboratory protocols, which provide textual descriptions of how to execute each phase of the experiment. Protocols play a fundamental role in ensuring reproducibility, allowing different laboratories to repeat procedures under comparable conditions and obtain reliable results [3]. Reproducibility has been widely recognized as a major challenge in modern experimental science, particularly in fields where complex procedures must be reproduced across laboratories and experimental settings.

In addition to protocols described in traditional scientific publications, a growing number of experimental procedures are shared through online protocol repositories and collaborative platforms. Resources such as protocols.io [24], OpenWetWare [25] host large collections of laboratory procedures covering a wide range of biological and biomedical experiments. These platforms facilitate knowledge sharing within the scientific community but also highlight the diversity and variability of protocol descriptions. While such repositories increase accessibility to experimental methods, they also amplify the challenges associated with standardizing and computationally interpreting natural-language protocols.

From a computational perspective, experimental protocols can be viewed as procedural texts that describe sequences of actions applied to physical materials under specific conditions. Similar to instructions in manufacturing processes or software workflows, laboratory protocols encode ordered operations, conditional steps, and parameter specifications. However, unlike formal programming languages, natural-language protocols often rely on implicit knowledge, domain-specific terminology, and contextual assumptions that are difficult to interpret automatically. For example, instructions such as "incubate briefly" or "wash thoroughly" may be meaningful for experienced researchers but remain ambiguous when interpreted computationally.

These characteristics make protocol understanding a challenging problem for automated systems. The interpretation of laboratory instructions requires identifying actions, materials, quantities, temporal constraints, and dependencies between steps. In many cases, the logical structure of the experiment must also be inferred, including loops, parallel operations, and conditional branches. Consequently, the transformation of natural-language protocols into structured procedural representations represents an important research problem at the intersection of computational biology and natural language processing.

However, protocols written in natural language often vary significantly in structure, level of detail, terminology, and completeness. Such variability can introduce ambiguity and lead to inconsistent outcomes across laboratories. Miss-

ing parameters, unclear instructions, or inconsistent terminology can affect experimental reliability and make computational interpretation extremely difficult [5,6]. As fabrication processes become more automated and technically complex, the need for clearer terminology, standardized documentation, and explicit procedural descriptions becomes increasingly important for the maturation of the biofabrication field [1, 2].

## 2.2 Structured Biological Protocols and Computational Representations

To address the limitations of free-text protocols, researchers have explored structured and machine-readable representations of biological procedures. Structured formats aim to formalize experimental workflows using standardized syntax, controlled vocabularies, and computationally interpretable descriptions. These representations make it possible to capture not only the sequence of experimental actions but also the associated parameters, dependencies, and conditions that define laboratory workflows.

One of the earliest examples is BioCoder, a domain-specific language designed to represent experimental workflows using explicit procedural operations and parameters [8]. BioCoder demonstrated that complex biological procedures can be formalized using constructs similar to those found in programming languages, improving clarity, reproducibility, and the potential for integration with laboratory automation systems. By explicitly defining actions such as reagent addition, incubation, and measurement operations, BioCoder illustrates how biological experiments can be expressed as structured computational workflows.

More recent work has extended these ideas toward more interoperable protocol representations. The Protocol Activity Markup Language (PAML) was introduced as an ontology-based framework for representing biological protocols and their execution records in a machine-readable form [4]. PAML builds on concepts from Unified Modeling Language (UML) activity diagrams, Autoprotocol, and Synthetic Biology Open Language (SBOL) Resource Description Framework (RDF) to describe protocol steps, control flow, data dependencies, and metadata in a unified format. Other efforts have proposed script-based protocol formats designed to facilitate laboratory automation and reproducible experimental workflows [26]. By defining a representation that bridges human-readable documentation and machine-executable instructions, such frameworks aim to improve reproducibility and interoperability across laboratories and automation environments.

Despite these advances, most structured protocol frameworks require protocols to be manually translated from natural-language descriptions into formal repre-

sentations [4,8]. Natural language processing research has also investigated structured representations of scientific procedures, for example through process-level annotation frameworks that capture actions, materials, and dependencies within experimental protocols [27]. Given the large number of experimental protocols available in scientific literature and online repositories, this manual conversion process limits scalability. Consequently, the main bottleneck is not the lack of representational frameworks but the absence of computational tools capable of automatically interpreting and structuring natural-language protocols [5,6].

## 2.3 Transformer Models and Large Language Models for Scientific Procedures

Recent advances in natural language processing have introduced new opportunities for addressing this challenge. The transformer architecture introduced by Vaswani et al. [9] reshaped the landscape of Natural Language Processing (NLP) by replacing recurrent architectures with an attention-based mechanism capable of modeling long-range dependencies in text.

The transformer architecture relies on a self-attention mechanism that allows each token in a sequence to attend to every other token in the input. This mechanism enables the model to capture long-range dependencies and contextual relationships more effectively than earlier recurrent or convolutional neural network architectures. In practice, a transformer model consists of stacked layers of multi-head self-attention modules followed by position-wise feedforward networks, allowing the model to learn hierarchical linguistic representations [9]. Positional encoding is used to preserve information about token order, ensuring that sequential relationships can still be modeled despite the parallel processing of tokens.

Another key characteristic of modern language models is the pretraining paradigm. Most large language models are first trained on large-scale text corpora using self-supervised learning objectives such as next-token prediction. This pretraining phase allows the model to acquire general linguistic knowledge and broad world knowledge before being adapted to downstream tasks. After pretraining, models can be further specialized using fine-tuning techniques on domain-specific datasets. This two-stage training strategy has proven highly effective for transferring general language understanding to specialized domains such as biomedical text processing.

One of the key advantages of transformer architectures is their ability to scale effectively with increasing model size and training data. Scaling laws observed in large language models indicate that model performance improves predictably as the number of parameters, training tokens, and computational resources increase [28, 29]. This property has enabled the development of models with billions of

parameters capable of capturing complex linguistic patterns and domain-specific knowledge. As a consequence, large language models have become powerful tools not only for general language tasks but also for specialized scientific applications.

Transformer architectures form the foundation of modern LLMs. Early models such as Bidirectional Encoder Representations from Transformers (BERT) introduced deep bidirectional representations that significantly improved contextual language understanding across a wide range of tasks [10]. Autoregressive models such as Generative Pre-trained Transformer (GPT)-3 later demonstrated that large-scale pretraining on extensive text corpora enables strong generative capabilities, allowing models to perform tasks such as summarization, question answering, and instruction following with minimal supervision [11]. Open foundation models such as LLaMA further expanded access to transformer-based architectures by demonstrating that high-quality language models can be trained efficiently using curated datasets and parameter-efficient techniques [12].

More recently, instruction tuning and parameter-efficient fine-tuning techniques have been introduced to adapt large language models to specific tasks without retraining the entire model. Methods such as Low-Rank Adaptation (LoRA) [30] enable efficient training by introducing small trainable parameter matrices while keeping the original pretrained weights frozen. This approach significantly reduces computational requirements while still allowing models to learn task-specific behaviors. Such techniques have become widely used in scientific and domain-specific applications where training datasets are limited but domain adaptation is required.

Alongside general-purpose models, a growing number of domain-adapted scientific language models have been developed to capture specialized terminology and knowledge present in scientific literature. BioGPT, for example, was trained on biomedical corpora and demonstrated strong performance on tasks such as biomedical relation extraction and terminology understanding [16]. Galactica extended this idea by training on a large scientific corpus containing articles, textbooks, molecular data, and mathematical expressions, enabling models to store and synthesize scientific knowledge across disciplines [17]. Surveys of scientific LLMs highlight the rapid development of such models and their increasing role in supporting scientific reasoning and information extraction [18]. More recent surveys further emphasize the growing role of large language models in scientific discovery, highlighting their use in literature synthesis, experimental reasoning, and knowledge extraction across multiple scientific disciplines [31].

Scientific language models differ from general-purpose language models in several important ways. Scientific texts often contain specialized terminology, structured argumentation, numerical information, and domain-specific symbols that are less common in general language corpora. As a result, models trained on general web-scale datasets may struggle to interpret scientific concepts accu-

rately. Domain-adapted models address this limitation by incorporating curated scientific corpora during training, allowing the models to better capture domain-specific semantics and relationships between scientific entities. Recent reviews further emphasize the transformative role of LLMs across multiple life-science domains, including genomics, proteomics, and drug discovery [32]. Similarly, recent studies highlight the growing opportunities for large language models in biomedical research, where they are increasingly used to analyze scientific literature and support knowledge discovery from complex biological data [33].

In the biomedical domain in particular, language models have been used for tasks such as named entity recognition, relation extraction, protein interaction prediction, and literature summarization. These applications demonstrate that language models can assist researchers in navigating the rapidly expanding volume of scientific publications. However, procedural understanding remains significantly more challenging than information extraction tasks, since it requires reasoning over ordered sequences of experimental actions and their dependencies. Early research in natural language processing has also investigated the automatic extraction of procedural information from wet-laboratory protocols. For example, Kulkarni et al. proposed methods for identifying experimental actions, reagents, and parameters from biological protocol text using machine learning and linguistic analysis [34]. Their work demonstrated that laboratory protocols can be interpreted as structured sequences of experimental operations, highlighting the feasibility of transforming natural-language procedures into computational representations. These early studies provide an important foundation for later work applying large language models to protocol understanding and generation.

Building on these capabilities, large language models are increasingly being explored as tools to support broader scientific discovery and research workflows. Beyond traditional natural language processing tasks, large language models are increasingly being explored as tools to support scientific discovery and research workflows. Recent studies have investigated their potential for tasks such as literature analysis, hypothesis generation, experimental planning, and knowledge extraction from scientific documents [18, 35]. In domains such as biology and chemistry, language models can assist researchers by synthesizing large volumes of scientific information and identifying relationships between experimental procedures, materials, and outcomes.

These capabilities have motivated the application of language models to procedural scientific text. Recent systems explore the use of LLMs to interpret and structure laboratory protocols. ProtoCode converts natural-language Polymerase Chain Reaction (PCR) protocols into structured operational instructions compatible with laboratory devices [20]. ProtoMed-LLM proposes evaluation methods for protocol formulation tasks using pseudocode representations and automated model assessment [36]. Similarly, BioPlanner introduces a dataset of biological

protocols paired with pseudocode representations and evaluates models on procedural planning tasks [19]. Also, another research investigates end-to-end systems where LLM agents assist in experimental design and scientific workflow automation [37].

Benchmark datasets further highlight the challenges of protocol understanding. BioLP-bench evaluates whether language models can detect failure-inducing modifications in biological workflows [5], while BioProBench provides a multi-task benchmark covering protocol question answering, step ordering, error correction, and procedural reasoning [6]. These studies demonstrate that although modern language models show promising capabilities, they still struggle with hierarchical procedural reasoning, step dependencies, and safety-critical details.

In parallel, research has begun exploring the integration of language models with automated laboratory systems. For example, Inagaki et al. showed that large language models can translate natural-language experiment descriptions into executable scripts for liquid-handling robots [38]. Similar approaches have been explored in chemistry through robotic platforms capable of executing complex experimental workflows guided by language models [39]. These developments point toward a broader role for LLMs in connecting natural-language scientific instructions with automated laboratory systems. More recently, LLM-based agents have been proposed to support biomedical research workflows by coordinating reasoning, data analysis, and experimental planning tasks within complex scientific pipelines [40].

The integration of language models with laboratory automation systems represents an emerging research direction in computational biology and experimental automation. These developments are also part of a broader movement toward integrating artificial intelligence, agent-based systems, and automated laboratory platforms to accelerate scientific discovery and experimental workflows [41]. By translating natural-language experiment descriptions into machine-executable instructions, language models may serve as an interface between human researchers and robotic laboratory platforms. Such systems could enable semi-automated experimental design, protocol verification, and automated execution of laboratory workflows. While these approaches remain in early stages of development, they illustrate how advances in natural language processing may fundamentally reshape the way experimental procedures are documented, shared, and executed.

## 2.4 Literature Gaps and Positioning of This Thesis

Although substantial progress has been made in biofabrication technologies, structured protocol representations, and scientific language models, several important

gaps remain. Biofabrication literature highlights the rapid growth of fabrication techniques and experimental complexity [1, 2], yet most protocols remain documented as unstructured natural-language text. This limits reproducibility, automation, and computational analysis of experimental workflows. Reviews of computational methods in biofabrication further highlight the lack of standardized procedural datasets and the need for frameworks that integrate protocols with modeling, simulation, and optimization tools [7, 42].

Several research efforts have therefore attempted to address this problem by introducing structured representations of laboratory protocols. Structured protocol frameworks such as BioCoder and PAML demonstrate that biological workflows can be formalized using machine-readable representations. However, these approaches typically require manual protocol encoding and therefore cannot easily scale to large collections of experimental documents. Recent LLM-based approaches show promise in interpreting procedural text, but they remain limited by the scarcity of domain-specific datasets and by difficulties in reasoning over complex multi-step workflows. Consequently, there is a clear need for scalable methods that can transform domain-specific experimental protocols into structured representations suitable for computational analysis and language model training.

In light of these challenges, this thesis is positioned at the intersection of biofabrication, NLP, and scientific automation. The work addresses the lack of scalable tools for transforming unstructured biofabrication protocols into structured, machine-readable representations.

Specifically, this thesis makes three main contributions:

1. the construction of a large-scale dataset of biofabrication protocols collected from heterogeneous scientific sources;

2. the development of a normalization pipeline that transforms unstructured procedural text into structured representations;

3. the adaptation and evaluation of transformer-based language models for generating structured biofabrication protocols.

Together, these contributions aim to support reproducible documentation and computational analysis of biofabrication workflows.

# Chapter 3

# METHODS

## 3.1 Methodological Overview

This work proposes a pipeline for transforming heterogeneous biofabrication protocols into structured representations for training large language models for protocol generation tasks. In addition to constructing a structured dataset, the proposed methodology also includes the fine-tuning and evaluation of a large language model capable of generating biofabrication protocols based on the extracted procedural structure. The methodology integrates multiple components, including data acquisition, protocol normalization, action extraction, dataset construction, model fine-tuning and validation.

The methodology is implemented through the following stages:

1. Collection of protocol-related documents
2. Raw Protocol Extraction
3. Construction of a laboratory action library
4. Transformation of protocols into a structured dataset
5. Fine-tuning of a large language model

Figure 3.1 illustrates the overall system architecture.

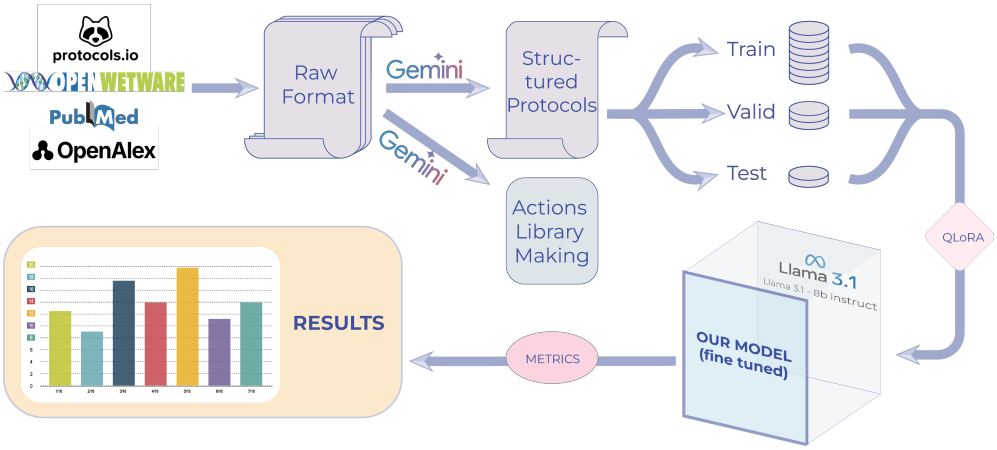


Figure 3.1: Overview of the proposed processing pipeline for biofabrication protocol generation. The process begins with the collection of experimental protocols from multiple scientific repositories such as protocols.io, OpenWetWare, PubMed, and OpenAlex. These heterogeneous documents are first gathered in their raw format and then processed using Gemini to transform the protocols into a normalized and structured representation. During this stage, an action library is also constructed to capture the standardized laboratory operations used across protocols. The resulting structured protocols are then organized into training, validation, and test datasets. Finally, the dataset is used to fine-tune a LLaMA 3.1 8B model using the QLoRA method, and the performance of the trained model is evaluated using multiple quantitative metrics.

Each stage of the pipeline is described in detail in the following sections.

## 3.2 Collection of Protocol-Related Documents

### 3.2.1 Data Sources, Search Query and Retriveal Procedure

Protocol-related documents were collected from four publicly available scientific resources frequently used to publish experimental methods:

- Protocols.io [24]

- OpenWetWare [25]
- OpenAlex [43]
- PubMed [44]

These platforms differ in the type of information they provide. Protocols.io [24] and OpenWetWare [25] are repositories specifically designed for sharing experimental protocols and laboratory workflows, often containing step-by-step procedural descriptions written by researchers. In contrast, PubMed [44] and OpenAlex [43] are large scientific literature databases that index peer-reviewed publications and associated metadata. While PubMed provides access to biomedical abstracts and links to full articles, OpenAlex offers structured metadata for scholarly publications and can be queried programmatically through an open API.

Because of these differences, the data extraction process varied across sources. Protocols.io [24] and OpenWetWare [25] were primarily used to retrieve detailed experimental protocols, which were obtained either through available APIs or through web scraping when programmatic interfaces were not available. PubMed [44] and OpenAlex [43] were instead used to identify scientific publications related to biofabrication and tissue engineering. From these sources, relevant documents and associated metadata were retrieved using their respective APIs.

To avoid redundancy in the collected dataset, duplicate entries were removed during the preprocessing stage. Duplicates were identified by comparing document titles, identifiers (such as DOIs), and textual similarity between retrieved protocol descriptions.

To retrieve relevant documents from these sources, keyword-based queries were used through the available APIs or web interfaces. The search query combined domain-specific terms related to biofabrication and tissue engineering with procedural keywords that typically indicate experimental protocols.

```
QUERY = (
Tissue engineering OR Biofabrication OR Regenerative medicine OR
Bioprinting OR Cell culture OR Biomedical engineering OR Biomaterials OR
3D bioprinting OR 4D bioprinting OR Organ-on-a-chip OR Stem cell
    technology OR
Synthetic biology OR Developmental biology OR Morphogenesis OR
    Ontogenesis OR
Organoid fabrication OR Differentiation OR Directed development OR
Transdifferentiation OR Spheroid Fabrication OR Scaffold Fabrication OR
Microphysiological system fabrication
) AND (Protocol OR blueprint OR process OR steps OR workflow)
```

To ensure consistency across data sources, the same logical query structure was used for all repositories. The query combines domain-specific keywords related

to biofabrication and tissue engineering with procedural terms that indicate the presence of experimental protocols.

Although the exact syntax required small adjustments depending on the specific API or search interface, the logical structure of the query remained the same for all sources. This approach ensured that the retrieved documents consistently contained both domain-related terminology and references to experimental procedures across the different repositories.

Documents returned by these searches were downloaded using APIs when available or through controlled web scraping procedures. Each retrieved document was stored as a JavaScript Object Notation (JSON) file containing the available metadata and textual content.

## 3.3 Raw Protocol Extraction

The downloaded documents contain heterogeneous information depending on the source platform. To standardize the dataset, a preprocessing stage was implemented to extract relevant protocol information.

During this stage, the following fields were extracted when available:

- **Protocol identifier**: an alphanumeric identifier assigned to each protocol entry in the source repository. These identifiers are unique within each individual repository, although different repositories may use different identifier formats and rules. Therefore, the combination of the repository source and the identifier is used to uniquely reference each collected protocol in the dataset.
- **Title**: a short textual description summarizing the protocol objective or experimental procedure.
- **Description**: an introductory section providing contextual information about the protocol, including its purpose or experimental background.
- **Protocol steps**: the ordered list of experimental steps describing the procedure to be performed.
- **Full text content**: the complete textual content of the protocol document, including descriptions, notes, and procedural instructions.

If a structured step list was available, it was used directly. Otherwise, the full protocol text was used as the raw procedural description.

The result of this stage is a collection of normalized raw protocol entries.

Example of a raw protocol entry extracted from the dataset:

```
....
....
 "Documents": null,
    "Steps": [
        {
            "Step Number": "1",
            "Step Text": "Grind frozen plant material (100mg) in liquid
               Nitrogen",
            "Tables": []
        },
        {
            "Step Number": "2",
            "Step Text": "In a 2 ml tube, add 800 uL of 1,5x CTAB and 1
               ul of Beta-mercaptoethanol to the ground leaf material
               Incubate 1 hour at 60-65 degrees (C)",
            "Tables": []
        },
        {
            "Step Number": "3",
            "Step Text": "Cool at Room Temp Add 1 volume of
               Chloroform/Isoamylalkohol(24:1) mixture Mix on overhead
               shaker for 10 minutes Centrifuge at 3000 rmp for 25
               minutes",
            "Tables": []
        },
        {
            "Step Number": "4",
            "Step Text": "Transfer supernatant to a new tube Use wide
               pipette tips (cut point of tips if needed)",
            "Tables": []
        },
....
....
```


Figure 3.2 illustrates the complete data construction pipeline.

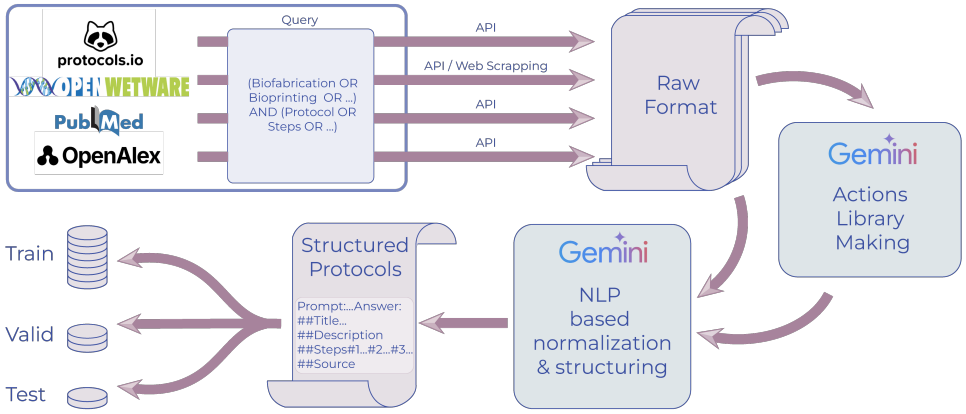


Figure 3.2: Data construction pipeline used to collect and preprocess protocol documents from multiple scientific repositories.

## 3.4 Construction of a Laboratory Action Library

A key component of the system is the creation of a library of laboratory actions that represent common operations performed in biological protocols.

To construct this library, the extracted raw protocol data was analyzed using the Gemini API. The raw protocol texts were provided to Gemini with prompts designed to identify recurring laboratory actions appearing in experimental procedures.

Since the initial list contained redundant or overly specific actions, a second filtering stage was performed. Gemini was again used to detect duplicated actions, merge semantically equivalent actions, and remove actions that were too specific to particular protocols. This library represents the standardized set of laboratory actions used throughout the dataset.

The functions in the action library also provide a partial implementation of how laboratory actions are normalized and represented. Each function defines a standardized interface for a specific experimental action (e.g., incubation, mixing, centrifugation) together with its possible parameters.

These implementations include conditional logic that converts heterogeneous textual descriptions into consistent procedural statements. For example, numerical values may be automatically converted into standardized units, optional pa-

rameters may be included when available, and descriptive text is generated to produce a uniform representation of the experimental step.

This standardization is important for the subsequent stages of the pipeline, as it ensures that different protocol descriptions referring to the same laboratory action are represented in a consistent format. As a result, the extracted steps can be reliably used to construct the structured protocol dataset and to train the language model.

Example entries from the action library:

```
....
....
def incubate(self,
                duration: Union[str, int, float],
                temperature: Optional[Union[str, int, float]] = None,
                item: Optional[str] = None,
                method: Optional[str] = None,
                shaking_rpm: Optional[Union[int, str]] = None,
                purpose: Optional[str] = None,
                agitation: Optional[bool] = None, # Alias for shaking
                comment: Optional[str] = None) -> None:
        """
        Incubates samples or items under specified conditions. Can also
            be used for thawing or baking.

        Args:
            duration: The duration of incubation (e.g., "30 minutes",
                2.5, "overnight", "4 hours", "until thawed").
            temperature: The temperature for incubation (e.g., "room
                temperature", 37, "4degreeC", "150degreeC", "ambient",
                "on ice").
            item: The item being incubated (optional, e.g., "cells",
                "slide", "mix", "brain tissue", "plasmid", "plate",
                "agarose gel", "virus vial").
            method: The incubation method (optional, e.g., "on
                thermocycler", "in oven", "shaker", "hotplate", "thermal
                cycling", "on rotator", "water bath").
            shaking_rpm: Optional shaking speed in RPM if applicable.
                Defaults to None.
            purpose: The reason for incubation (e.g., "thawing",
                "baking", "growth", "polymerization", "for lysis",
                "enzymatic digestion"). Defaults to None.
            agitation: Whether to agitate during incubation
                (True/False). Overrides ‘shaking_rpm‘ if True and
                ‘shaking_rpm‘ is None. Defaults to None.
```

```
    comment: Additional comments or context. Defaults to None.
"""
duration_str = str(duration)
if isinstance(duration, (int, float)):
    # Heuristic for duration unit based on magnitude.
    if duration >= 10: # Assume minutes or hours if large, but
        prioritize minutes for typical lab incubations
        duration_str = "{duration}
            minutes".format(duration=duration)
    elif duration >= 1: # Assume minutes if moderate
        duration_str = "{duration}
            minutes".format(duration=duration)
    else: # Assume seconds if small float
        duration_str = "{duration:.0f}
            seconds".format(duration=duration * 60)

temp_str = str(temperature)
if isinstance(temperature, (int, float)):
    temp_str =
        "{temperature}degreeC".format(temperature=temperature)

description_parts = ["Incubating"]
if item:
    description_parts.append(item)
if purpose:
    description_parts.append("for
        {purpose}".format(purpose=purpose))

description_parts.append("for
    {duration_str}".format(duration_str=duration_str))

if temperature is not None:
    description_parts.append("at
        {temp_str}".format(temp_str=temp_str))
if method:
    description_parts.append("using
        {method}".format(method=method))

if shaking_rpm is not None:
    description_parts.append("with shaking at {shaking_rpm}
        rpm".format(shaking_rpm=shaking_rpm))
elif agitation: # Fallback for boolean agitation
    description_parts.append("with agitation")
```

```
        if comment:
            description_parts.append("({comment})".format(comment=comment))
        self._log_action(" ".join(description_parts).strip() + ".")
....
....
```

In total, the action library contains 213 functions covering common laboratory procedures such as incubation, mixing, centrifugation, washing, and measurement.

## 3.5 Transformation of Protocols into a Structured Dataset

### 3.5.1 Structured Protocol Generation

Once the action library was created, the raw protocol dataset was processed again using the Gemini API.

For each protocol entry, Gemini was prompted to transform the raw text into a structured format consisting of:

- **Protocol prompt**: the textual instruction provided to the language model, describing the objective of the protocol to be generated.
- **Protocol title**: a short phrase summarizing the purpose or name of the experimental procedure.
- **Protocol description**: a brief explanatory text providing context about the experimental goal, materials, or biological process involved.
- **Structured procedural steps**: an ordered sequence of experimental actions representing the protocol workflow, where each step is expressed using standardized actions defined in the action library.

Each step in the resulting protocol consists of an action selected from the predefined action library along with its associated parameters.

Example of a structured protocol representation:

```
{"question": "Please create a protocol in order to extract DNA from
    ethanol-preserved zooplankton samples using phenol-chloroform
    extraction.",
"answer": "
# Title: DNA Extraction from Ethanol Zooplankton Samples -
    Phenol-Chloroform
\n# Description: This protocol outlines the steps for extracting
    genomic DNA from zooplankton samples preserved in ethanol,
    utilizing the phenol-chloroform method.
```

```
\n#Steps:
\n#1 add_reagent(reagent_name='lysozyme', volume='100 uL', stock='125
   mg in 1000 ul 1 x TE')
\n#2 add_reagent(reagent_name='RNase A', volume='20 uL', stock='10
   ug/ml')
\n#3 incubate(temperature='37degreeC', time='1h', shaking='true')
\n#4 add_reagent(reagent_name='Proteinase K', volume='100 uL')
\n#5 add_reagent(reagent_name='SDS', volume='100 uL',
   concentration='20%')
\n#6 incubate(temperature='55degreeC', time='1-2h', shaking='true')
\n#7 add_reagent(reagent_name='sucrose lysis buffer (SLB)',
   volume='1mL')
\n#8 add_reagent(reagent_name='Phenol:Chloroform:IAA (25:24:1)',
   volume='3mL', pH='8.0')
\n#9 mix_and_transfer_reagents(mix_method='invert',
   inversion_count='10s')
\n#10 centrifuge(speed='2500 g', time='6 min')
\n#11 transfer_item(item_name='aqueous layer', destination='new 15 ml
   falcon tube')
\n#12 add_reagent(reagent_name='Chloroform:IAA (24:1)', volume='3mL')
\n#13 mix_and_transfer_reagents(mix_method='invert',
   inversion_count='10s')
\n#14 centrifuge(speed='2500 g', time='6 min')
\n#15 transfer_item(item_name='aqueous layer', destination='Amicon
   Ultra centrifuge tube')
\n#16 add_reagent(reagent_name='1 x TE buffer', volume='1-2mL')
\n#17 centrifuge(speed='3500 g', time='10-15 min')
\n#18 add_reagent(reagent_name='TE buffer', volume='2 mL')
\n#19 centrifuge(speed='3500 g', time='6 min')
\n#20 remove_item(item_name='filtrate')
\n#21 add_reagent(reagent_name='1xTE', volume='50 uL')
\n#22 transfer_item(item_name='sample in Amicon', destination='1.5 mL
   Eppendorf tube')
\n#23 aliquot_and_store(sample_volume='50 uL',
   storage_temperature='-20degreeC', storage_type='working stock')
\n#24 store_item(storage_temperature='-80degreeC',
   storage_type='long-term stock')
\n#25 input_analysis_details()
\n#26 aliquot_and_store(sample_volume='25 uL',
   target_concentration='2.5 ng/uL', purpose='PCR stock')
\n#27 store_item(storage_temperature='-20degreeC', storage_type='DNA
   stock')
\n#28 weigh_and_transfer_reagent(reagent_name='Sucrose Lysis Buffer
   (SLB)', volume='1800 uL')
```

```
\n#29 add_reagent(reagent_name='homogenized zooplankton sample',
   volume='100 uL')
\n#30 add_powder_to_liquid(powder='Stainless Steel UFO Beads',
   quantity='3', size='3.5 mm')
\n#31 add_powder_to_liquid(powder='Stainless Steel Beads',
   quantity='30', size='0.9 - 2.0 mm blend')
\n#32 sonicate_cells(duration='10 min')
\n#33 aliquot_and_store(sample_volume='50 uL',
   storage_temperature='-20degreeC', storage_type='working stock')
\n#34 store_item(storage_temperature='-80degreeC',
   storage_type='long-term stock')
\n#35 input_analysis_details()
\n#36 aliquot_and_store(sample_volume='25 uL',
   target_concentration='2.5 ng/uL', purpose='PCR stock')
\n#37 store_item(storage_temperature='-20degreeC', storage_type='DNA
   stock')",
"source":
   "/root/biofabrication-data-collector/protocols/protocol-io/protocols_versions_jso
{"question": "Please create a protocol in order to synthesize mRNA and
   purify it.",
"answer":
"# Title: mRNA Synthesis and Purification Protocol
\n# Description: This protocol outlines the steps for synthesizing mRNA
   using the HiScribe T7 mRNA Kit with CleanCap Reagent AG, followed
   by purification using the Monarch RNA Cleanup Kit.
\n# Steps:
\n#1 add_reagents_to_tubes(reagents='Reaction Buffer, NTPs, CleanCap
   Reagent AG, Template DNA', volumes='12 uL each', conditions='room
   temperature, thaw, mix, microfuge')
\n#2 incubate(temperature='37 degreeC', duration='2 hours')
\n#3 add_reagent(reagent='DNase I', volume='2 uL')
\n#4 incubate(temperature='37 degreeC', duration='15 minutes')
\n#5 dilute_solution(solution='reaction mixture',
   diluent='nuclease-free water', final_volume='50 uL')
\n#6 add_reagents_to_tubes(reagents='Cleanup Binding Buffer',
   volume='100 uL')
\n#7 add_reagent(reagent='ethanol (>= 95%)', volume='150 uL')
\n#8 centrifuge(duration='1 minute')
\n#9 wash_samples(wash_buffer='Cleanup Wash Buffer', volume='500 uL')
\n#10 centrifuge(duration='1 minute')
\n#11 wash_samples(wash_buffer='Cleanup Wash Buffer', volume='500 uL')
\n#12 centrifuge(duration='1 minute')
\n#13 elute(eluent='nuclease-free water', volume='20 uL')",
"source":
```

"/root/biofabrication-data-collector/protocols/protocol-io/protocols_versions_json/10096

The structured protocols were then stored in an JSON Lines (JSONL) dataset where each line corresponds to a single protocol entry.

Figure 3.3 shows an example of the transformation from raw protocol text to structured protocol representation.

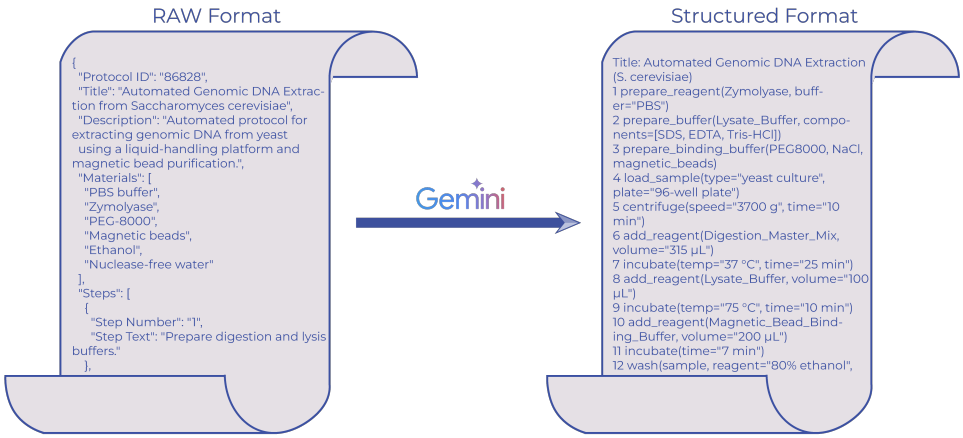


Figure 3.3: Example of a protocol representation before and after the normalization process. The left side shows the original raw protocol format containing unstructured metadata and step descriptions, while the right side illustrates the structured protocol representation generated through the NLP-based normalization process and the action library.

### 3.5.2 Dataset Preparation

The resulting dataset was divided into three subsets used for model training and evaluation:

- Training set (70%)
- Validation set (20%)
- Test set (10%)

Each dataset entry is constructed as an instruction-response pair. The instruction corresponds to the textual description of the protocol, which acts as the

input prompt for the model. The response contains the structured procedural representation of the protocol.

In this representation, laboratory steps are expressed using the standardized functions defined in the action library. These functions serve as a gold standard representation of laboratory actions, ensuring that each experimental step is described using a consistent set of operations and parameters.

Each dataset entry is constructed as an instruction-response pair. The instruction corresponds to the textual description of the protocol, which acts as the input prompt for the model, while the response contains the structured procedural representation of the protocol. In this representation, laboratory steps are expressed using the standardized functions defined in the action library. These functions provide a consistent representation of laboratory actions and therefore act as a reference structure for the correct procedural output. During training, the model learns to map natural language protocol descriptions to their corresponding sequence of standardized laboratory actions.

### 3.5.3 Tokenization and Context Length Analysis

Before training the language model, the dataset was tokenized to analyze the length distribution of protocol entries.

Token statistics were computed for each protocol in the dataset to analyze the distribution of sequence lengths and determine an appropriate maximum context length for the language model during training. The distribution of token counts was visualized using a histogram, which allowed analysis of the typical protocol length within the dataset.

Figure 3.4 shows the token length distribution across the dataset.

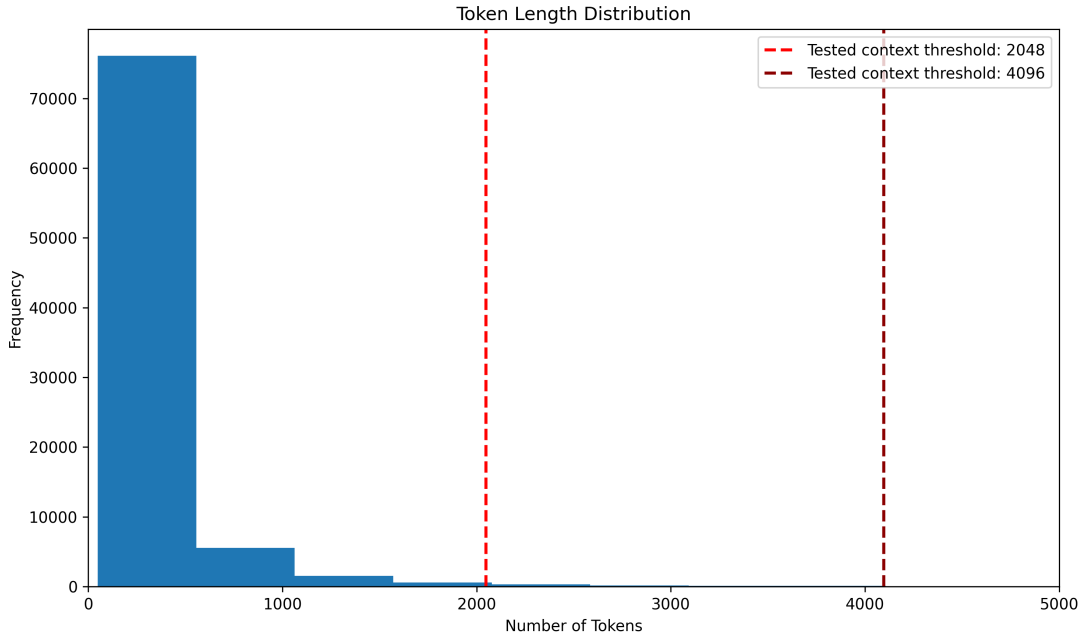


Figure 3.4: Token length distribution of protocols in the training dataset after tokenization. The histogram illustrates the frequency of protocols across different token length ranges. Most protocols are relatively short, resulting in a strong concentration in the lower token ranges. The dashed vertical lines represent the context length thresholds tested during model configuration (2048 and 4096 tokens). These thresholds indicate the maximum sequence lengths considered during training and are shown as reference markers rather than observed data points.

The token length distribution analysis shows that most protocols are relatively short. The dataset contains 84,873 samples with an average length of 379 tokens and a median length of 248 tokens. Furthermore, 90% of protocols contain fewer than 566 tokens and 95% contain fewer than 865 tokens. Only a small fraction of samples exceed the tested context thresholds: 1.33% of protocols are longer than 2048 tokens and 0.50% exceed 4096 tokens.

These statistics confirm that a context length of 2048 tokens is sufficient to capture the full content of the vast majority of protocols while keeping computational requirements manageable during training.

Based on this analysis, a context length of **2048 tokens** was selected for model training. This value covers the majority of protocol examples while maintaining efficient training performance.

## 3.6 Fine-Tuning of a Large Language Model

The final stage of the pipeline consists of fine-tuning a large language model on the constructed dataset.

Figure 3.5 illustrates the fine-tuning workflow.

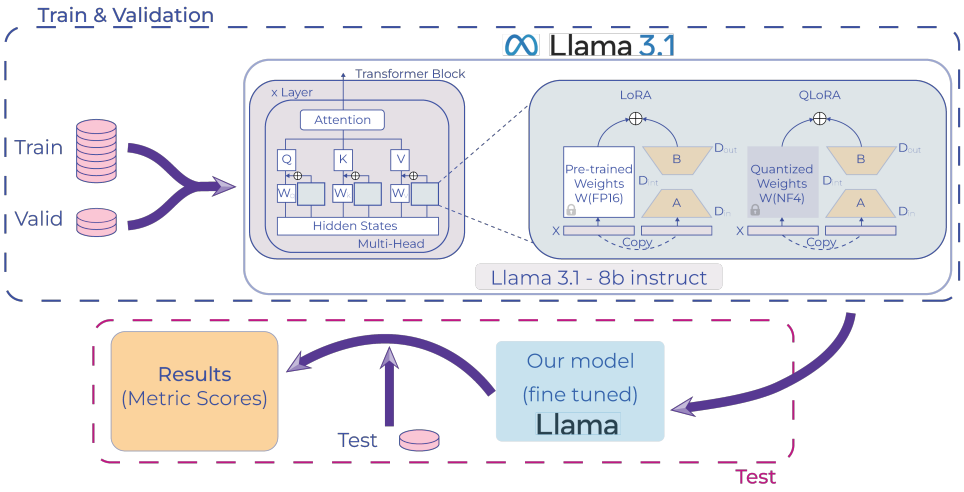


Figure 3.5: Overview of the model fine-tuning and evaluation pipeline. The process begins with the structured protocol dataset, which is divided into training and validation subsets used during the fine-tuning stage. The model is based on LLaMA 3.1 8B Instruct, where the transformer blocks remain frozen and parameter-efficient adaptation is performed using the QLoRA approach. During training, low-rank adaptation matrices are introduced to update the model behavior while keeping the original pretrained weights fixed. After the fine-tuning stage, the resulting model is evaluated using the held-out test dataset. The generated protocols are then compared with reference protocols using multiple evaluation metrics to produce the final performance results.

Large language models have recently demonstrated strong performance in tasks that require structured text generation and reasoning over procedural descriptions. In particular, instruction-tuned models are well suited for tasks where the input describes an objective and the output consists of a structured response. Since the dataset constructed in this work follows an instruction-response format, fine-tuning an instruction-following language model provides a natural framework for learning structured protocol generation.

The model used in this work is **LLaMA 3.1 8B Instruct**, a decoder-only transformer-based language model developed by Meta [45]. The model contains approximately eight billion parameters and is optimized for instruction-following tasks through supervised fine-tuning. Compared to larger models, the 8B configuration provides a favorable balance between model capability and computational requirements, which makes it suitable for experimentation and parameter-efficient fine-tuning approaches.

### 3.6.1 Language Modeling Objective

The fine-tuning process follows the standard autoregressive language modeling objective used for decoder-only transformer models. Given a sequence of tokens $x = (x_1, x_2, ..., x_T)$, the model estimates the probability of each token conditioned on the tokens that precede it:

$$P(x) = \prod_{t=1}^{T} P(x_t | x_{<t}) \tag{3.1}$$

where $x_{<t}$ denotes the sequence of tokens preceding position $t$. During training, the model parameters are optimized by minimizing the negative log-likelihood loss:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(x_t | x_{<t}) \tag{3.2}$$

This objective encourages the model to assign high probability to the correct next token in the training sequence.

In the instruction fine-tuning setting used in this work, each example consists of an instruction sequence $I$ and a response sequence $R$. During training, the model receives the instruction and the previous response tokens as context and is trained to predict the next token of the ground-truth response. The objective therefore becomes:

$$\mathcal{L} = -\sum_{t=1}^{|R|} \log P(r_t | I, r_{<t}) \tag{3.3}$$

where $r_t$ denotes the the $t$-th token of the reference response. Minimizing this loss encourages the model to assign high probability to the correct tokens of the structured protocol given the input instruction.

This training strategy corresponds to the standard teacher-forcing paradigm commonly used for training autoregressive language models [46].

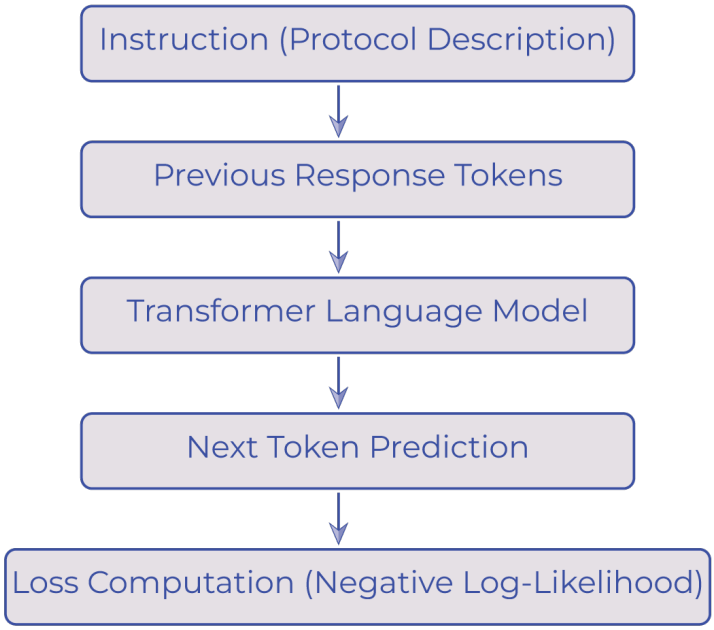


Figure 3.6: Autoregressive training process used during instruction fine-tuning. The model receives the instruction and previous response tokens as context and is trained to predict the next token of the reference response.

### 3.6.2 Parameter-Efficient Fine-Tuning with QLoRA

To efficiently fine-tune the model on the protocol dataset, the QLoRA [21]technique was used. QLoRA is a parameter-efficient fine-tuning method that combines low-rank adaptation (LoRA) [30] with model quantization.

In LoRA, instead of updating the full pretrained weight matrix $W \in \mathbb{R}^{d\times k}$, the weight update is approximated using two low-rank matrices:

$$\Delta W = BA \tag{3.4}$$

where

$$B \in \mathbb{R}^{d\times r}, \quad A \in \mathbb{R}^{r\times k}$$

and $r$ is the rank of the low-rank adaptation with $r \ll \min(d, k)$.
The adapted weight matrix becomes:

$$W' = W + BA \tag{3.5}$$

During training, only the matrices $A$ and $B$ are updated while the original pretrained weights $W$ remain frozen. This significantly reduces the number of trainable parameters.

The low-rank matrices $A$ and $B$ are initialized at the beginning of training rather than computed from a decomposition of $W$. Typically, $A$ is initialized with small random values while $B$ is initialized to zero. This initialization ensures that the initial weight update $\Delta W = BA$ is close to zero, so the model initially behaves like the original pretrained model. During fine-tuning, the matrices $A$ and $B$ are optimized through gradient descent while the pretrained weights $W$ remain frozen.

This approach significantly reduces the number of trainable parameters while preserving most of the representational capacity of the original model.

QLoRA [21] further improves memory efficiency by quantizing the base model weights into 4-bit representations,following the configuration proposed in the original QLoRA framework which employs NormalFloat4 (NF4) quantization, while keeping the LoRA [21] adapter parameters in higher precision during training. This allows large models such as LLaMA 3.1 8B to be fine-tuned on limited GPU resources while maintaining competitive performance.

### 3.6.3 Training Configuration

The main training configuration used in this work includes:

| **Parameter** | **Value** |
|---|---|
| Model | LLaMA 3.1 8B Instruct |
| Fine-tuning method | QLoRA |
| Batch size | 2 |
| Maximum sequence length | 2048 tokens |
| Training epochs | 20 |
| Optimizer | AdamW optimizer |

Table 3.1: Training configuration used for model fine-tuning.

Early stopping was applied during training to reduce the risk of overfitting. The stopping criterion was based on the validation loss computed during periodic evaluations on the validation dataset. Validation was performed every 500 training iterations, resulting in periodic validation evaluations throughout the training

process. The patience parameter was set to five evaluations. Therefore, if the validation loss did not decrease for five consecutive validation evaluations (i.e., 2500 training iterations), the training process was automatically terminated. In other words, the training continued only while improvements in validation performance were observed. The model parameters corresponding to the best validation loss were retained as the final trained model.

The sequence length of 2048 tokens was selected based on the token length analysis described in the previous section. Early stopping was applied to prevent overfitting during training.

### 3.6.4 Training Metrics and Monitoring

During training, multiple metrics were monitored to evaluate model performance. The primary training objective was the language modeling loss defined in Equation (3). In addition to the loss value, several auxiliary metrics were tracked during training and evaluation.

#### Mean Token Accuracy

The *mean token accuracy* measures the proportion of tokens predicted correctly by the model:

$$\text{Accuracy} = \frac{\text{Number of correctly predicted tokens}}{\text{Total number of tokens}} \tag{3.6}$$

#### Prediction Entropy

Entropy measures the uncertainty of the probability distribution predicted by the model over the vocabulary. For a probability distribution $p(x)$ over possible tokens, the entropy is defined as:

$$H(p) = -\sum_{i=1}^{V} p(x_i) \log p(x_i) \tag{3.7}$$

where $V$ is the size of the vocabulary and $p(x_i)$ is the probability assigned by the model to token $x_i$.

Higher entropy indicates that the probability mass is distributed across many possible tokens, reflecting greater uncertainty in the model predictions. Conversely, lower entropy indicates that the model assigns higher probability to a small number of tokens, corresponding to more confident predictions.

During training, both training loss and validation loss were monitored in order to track the learning dynamics of the model and detect potential overfitting.

### 3.6.5 Evaluation Metrics

Evaluation metrics were computed after model training to assess the quality of the generated outputs on the validation and test datasets. These metrics are distinct from the training metrics described in the previous section, which were used to monitor the optimization process during fine-tuning.

Since protocol generation involves both natural language generation and structured procedural correctness, multiple complementary metrics were used to evaluate lexical similarity, semantic similarity, structural correctness, and generation diversity.

All metrics were computed by comparing the generated sequence $G$ with the corresponding reference sequence $R$.

#### N-gram Based Metrics

N-gram based metrics measure lexical similarity between generated and reference sequences. An *n-gram* is a contiguous sequence of $n$ tokens extracted from a text. For example, in the sequence "cell culture medium", the bigrams (2-grams) are "cell culture" and "culture medium". These metrics evaluate how many such token sequences overlap between the generated output and the reference sequence.

#### BLEU

Bilingual Evaluation Understudy (BLEU) measures the precision of n-gram overlaps between the generated sequence and the reference sequence:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.8}$$

where:

- $p_n$ is the modified precision of n-grams of length $n$
- $w_n$ is the weight associated with each n-gram order
- $BP$ is the brevity penalty used to penalize overly short generated sequences
- $N$ is the maximum n-gram order considered

In this work, BLEU-4 was used, meaning that n-grams up to length $n = 4$ were considered.

### ROUGE-L

Recall-Oriented Understudy for Gisting Evaluation (ROUGE)-L evaluates similarity based on the longest common subsequence (LCS) between the generated sequence $G$ and the reference sequence $R$.

$$R_{LCS} = \frac{LCS(G, R)}{|R|} \tag{3.9}$$

$$P_{LCS} = \frac{LCS(G, R)}{|G|} \tag{3.10}$$

$$F1 = \frac{(1 + \beta^2) P_{LCS} R_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \tag{3.11}$$

where:

- $LCS(G, R)$ is the length of the longest common subsequence between $G$ and $R$
- $|G|$ is the length of the generated sequence
- $|R|$ is the length of the reference sequence

### Semantic Similarity Metrics

Lexical overlap metrics may fail to capture semantic similarity when the same procedure is expressed using different wording. Therefore, semantic similarity metrics were also used.

### BERTScore

BERTScore evaluates similarity using contextual embeddings produced by a pretrained transformer model.

Given token embeddings $x_i$ from the generated sequence and $y_j$ from the reference sequence, their similarity is computed using cosine similarity:

$$s(x_i, y_j) = \frac{x_i \cdot y_j}{\|x_i\| \|y_j\|} \tag{3.12}$$

where:

- $x_i$ is the embedding vector of the $i$-th token in the generated sequence
- $y_j$ is the embedding vector of the $j$-th token in the reference sequence

- $\|\cdot\|$ denotes the Euclidean norm

Precision, recall, and F1 are then computed using these similarity scores across all tokens.

### Protocol Structure Metrics

Since the goal of this work is to generate structured biofabrication protocols, evaluation must also consider the correctness of the procedural structure.

### Step-Level Metrics

Step-level metrics evaluate how well the generated protocol reproduces the sequence of procedural steps in the reference protocol.

First, both generated and reference protocols are divided into steps using step markers such as numbered lines, bullet points, or similar list-based formatting. Each extracted step is treated as a textual unit representing one protocol action or instruction.

To compare generated and reference steps, a fuzzy matching procedure is used rather than exact string matching. The similarity between a generated step and a reference step is computed using token-set similarity. Based on these pairwise similarities, a greedy matching procedure is applied so that each generated step can be matched to at most one reference step, and each reference step can be matched to at most one generated step. A step pair is considered matched only if its similarity score exceeds a predefined threshold.

After step matching, step-level precision, recall, and F1 are computed as:

$$Precision_{step} = \frac{N_{match}}{N_{pred}} \tag{3.13}$$

$$Recall_{step} = \frac{N_{match}}{N_{ref}} \tag{3.14}$$

$$F1_{step} = \frac{2 \cdot Precision_{step} \cdot Recall_{step}}{Precision_{step} + Recall_{step}} \tag{3.15}$$

where:

- $N_{match}$ is the number of matched step pairs
- $N_{pred}$ is the number of generated steps
- $N_{ref}$ is the number of reference steps

In addition to these metrics, three auxiliary step-level indicators are also computed:

- **Step Match Average Similarity**, which measures the average similarity score of matched step pairs
- **Step Count Ratio**, which compares the number of generated steps with the number of reference steps
- **Step Order LCS**, which measures how well the order of generated step actions follows the reference order using the longest common subsequence

### Action-Level Metrics

Action-level metrics evaluate whether the generated protocol contains the expected experimental action types.

After extracting protocol steps, the action associated with each step is identified using the first lexical item of the step text. This token is treated as a simplified representation of the main action performed in that step. Examples include action verbs or symbolic operation names such as `incubate`, `mix`, or `centrifuge`.

Let $A_G$ denote the set of action labels extracted from the generated protocol and $A_R$ the set of action labels extracted from the reference protocol. Action-level precision, recall, and F1 are then computed as:

$$Precision_{action} = \frac{|A_G \cap A_R|}{|A_G|} \tag{3.16}$$

$$Recall_{action} = \frac{|A_G \cap A_R|}{|A_R|} \tag{3.17}$$

$$F1_{action} = \frac{2 \cdot Precision_{action} \cdot Recall_{action}}{Precision_{action} + Recall_{action}} \tag{3.18}$$

This metric measures whether the generated protocol recovers the main action types present in the reference protocol, independently of exact wording or full step-level correspondence.

### Parameter-Level Metrics

Parameter-level metrics evaluate whether the generated protocol contains the experimental parameters present in the reference protocol.

Parameters are extracted from both generated and reference texts using regular expressions that detect numeric values followed by common laboratory units.

The extraction process captures expressions representing quantities such as temperature, time, volume, mass, and percentages. Examples include patterns such as `37 °C`, `10 min`, or `500 µL`.

Let $P_G$ denote the set of parameters extracted from the generated protocol and $P_R$ the set extracted from the reference protocol. Parameter-level precision, recall, and F1 are computed as:

$$Precision_{param} = \frac{|P_G \cap P_R|}{|P_G|} \tag{3.19}$$

$$Recall_{param} = \frac{|P_G \cap P_R|}{|P_R|} \tag{3.20}$$

$$F1_{param} = \frac{2 \cdot Precision_{param} \cdot Recall_{param}}{Precision_{param} + Recall_{param}} \tag{3.21}$$

This metric evaluates whether the generated protocol preserves the key experimental parameters required to reproduce the procedure.

### Token-Level Similarity

Token F1 evaluates lexical overlap between the generated sequence and the reference sequence at the token level.

Before computing the metric, both sequences are tokenized using a lightweight rule-based tokenizer. The tokenizer converts the text to lowercase and separates words and punctuation symbols using regular expressions. This approach provides a simple and model-independent token representation.

Let $T_G$ denote the set of tokens in the generated sequence and $T_R$ the set of tokens in the reference sequence. Token-level precision, recall, and F1 are computed as:

$$Precision_{token} = \frac{|T_G \cap T_R|}{|T_G|} \tag{3.22}$$

$$Recall_{token} = \frac{|T_G \cap T_R|}{|T_R|} \tag{3.23}$$

$$F1_{token} = \frac{2 \cdot Precision_{token} \cdot Recall_{token}}{Precision_{token} + Recall_{token}} \tag{3.24}$$

This metric provides a simple lexical similarity measure that is independent of the model's internal tokenization.

### Exact Match

Exact Match evaluates whether the generated sequence exactly matches the reference sequence:

$$EM = \begin{cases} 1 & \text{if } G = R \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

### Structural Validity

Structural Validity evaluates whether the generated output roughly follows the expected structural format of the protocol representation.

In this work, structural validity is implemented as a lightweight heuristic check. A generated protocol is considered structurally valid if:

- the output contains numbered protocol steps (e.g., `#1`, `#2`)
- parentheses used in action expressions are balanced

The metric is therefore defined as a binary indicator:

$$SV = \begin{cases} 1 & \text{if the output satisfies the structural checks} \\ 0 & \text{otherwise} \end{cases} \tag{3.26}$$

### Distinct-n

Distinct-n measures the proportion of unique n-grams in generated outputs:

$$Distinct_n = \frac{|\text{Unique } n\text{-grams in } G|}{|\text{Total } n\text{-grams in } G|} \tag{3.27}$$

where $|\cdot|$ denotes the number of elements in the set. Distinct-n metrics are computed at the sample level for each generated protocol and then averaged across the evaluation dataset.

### Length Ratio

Length Ratio compares the length of the generated sequence with the reference sequence:

$$Length\ Ratio = \frac{|G|}{|R|} \tag{3.28}$$

where $|G|$ and $|R|$ represent the token lengths of the generated and reference sequences respectively.

Table 3.2: Summary of evaluation metrics used to assess the quality of generated outputs.

| Metric | Purpose |
|---|---|
| BLEU-4 | N-gram lexical similarity |
| ROUGE-L | Longest common subsequence based similarity |
| BERTScore-F1 | Semantic similarity |
| Step Recall | Recall of action names extracted from numbered protocol steps |
| Exact Match (EM) | Exact protocol match |
| Token F1 | Token-level overlap similarity |
| Structural Validity | Heuristic structural format check |
| Distinct-2 / Distinct-3 | Output diversity |
| Length Ratio | Output length comparison |

# Chapter 4

# RESULTS AND EVALUATION

This chapter presents the experimental results obtained from fine-tuning the language model on the normalized biofabrication protocol dataset. The objective of the evaluation is to determine whether domain-specific training improves the ability of the model to generate structured and procedurally consistent laboratory protocols.

The evaluation compares the original pretrained model with the fine-tuned model across multiple quantitative metrics designed to capture both textual similarity and procedural correctness. In addition, qualitative examples are analyzed to illustrate the behavioral differences between the two models when generating experimental protocols.

## 4.1 Training Dynamics

The training dynamics of the fine-tuning process are illustrated in Figure 4.1. The figure reports the evolution of training and evaluation loss, training and evaluation token accuracy, and selected optimization variables over the course of training. The loss corresponds to the autoregressive language modeling objective defined in Section 3, which minimizes the negative log-likelihood of the reference response tokens conditioned on the input instruction and the previously generated tokens.

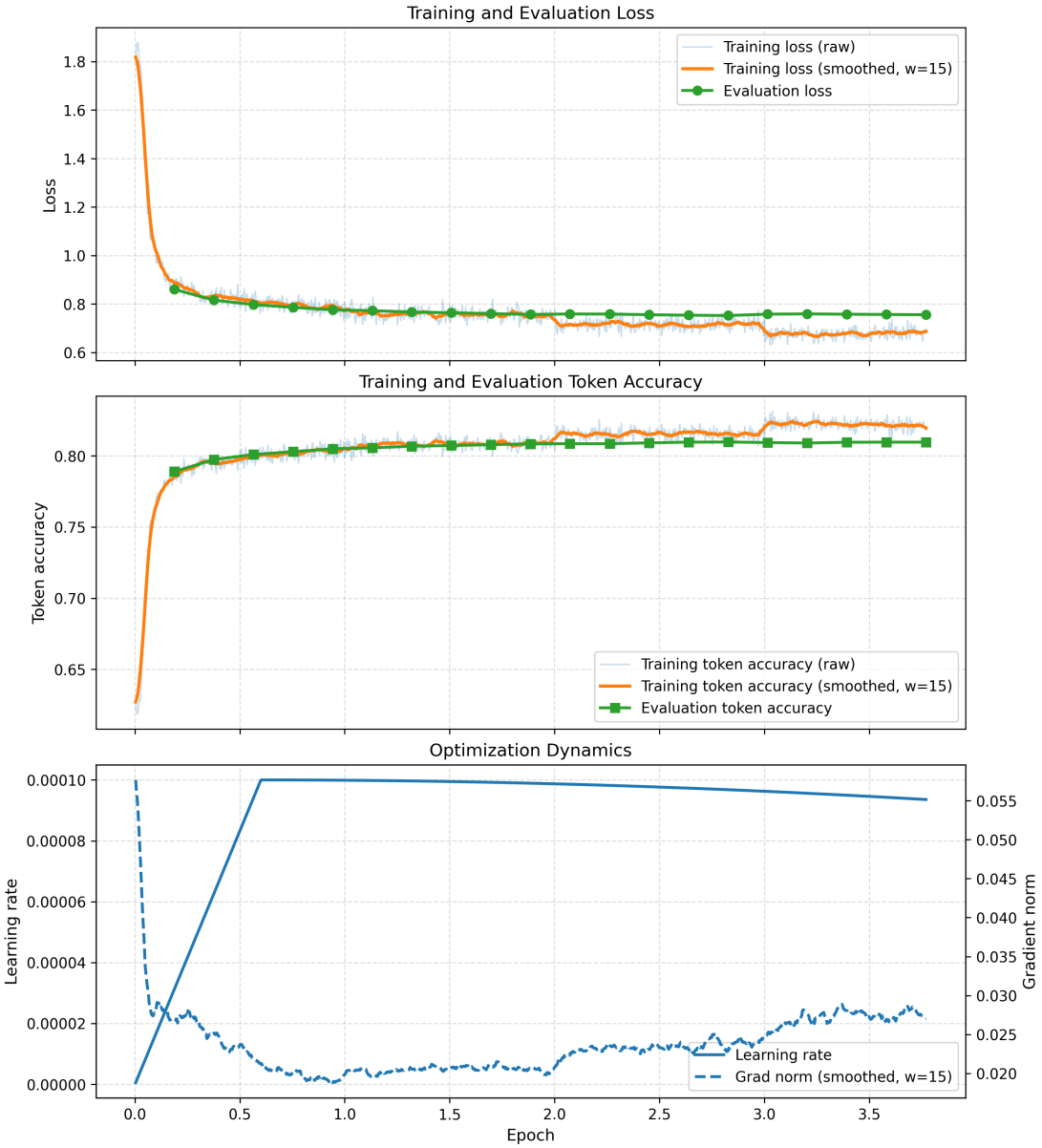


Figure 4.1: Training dynamics observed during fine-tuning of the LLaMA 3.1 8B model on the structured protocol dataset. The figure shows the evolution of loss, token accuracy, learning rate, and gradient norm across training epochs.

Several observations can be drawn from Figure 4.1. First, both training and evaluation loss decrease rapidly during the initial phase of training, indicating that the model quickly adapts to the structured protocol format. After this initial phase, the loss curves continue to decrease more gradually and eventually stabilize,

suggesting convergence toward a consistent solution.

Second, the evaluation loss remains close to the training loss throughout the training process and does not show a strong upward trend. This behavior suggests that the model does not exhibit strong overfitting. In typical overfitting scenarios, the training loss continues to decrease while the validation loss starts to increase, indicating that the model is specializing excessively on the training data and generalizing poorly to unseen samples. In this case, the similar trends observed for training and evaluation loss suggest that the model maintains a reasonable level of generalization during training. The best validation loss is reached near the selected checkpoint, after which further improvements become marginal. During training, an early stopping mechanism was used to monitor the validation loss and automatically select the best-performing checkpoint. The checkpoint selected by early stopping corresponds to the model with the lowest validation loss, which in this experiment was obtained at training step 7500. This checkpoint was therefore used as the final model for evaluation on the test set.

A similar trend is observed for token accuracy (see Section 3 for the definition of this metric). Training token accuracy increases from approximately 0.62 at the beginning of training to above 0.82, while evaluation token accuracy rises to slightly above 0.80 and then stabilizes. The relatively small gap between training and evaluation token accuracy suggests that the learned patterns are not limited to the training data and generalize reasonably well to unseen validation examples.

The optimization curves also support the stability of the training process. The learning rate follows the expected warm-up and decay schedule, while the gradient norm decreases sharply during the early phase and then remains within a relatively stable range. This suggests that the optimization process remains well-behaved and does not suffer from severe instability during fine-tuning.

Although training was configured for up to 20 epochs, early stopping terminated the training earlier when the validation loss stopped improving. Training progress was monitored in terms of optimization steps, where each step corresponds to one batch update of the model parameters. Given the size of the training dataset and the batch configuration, each epoch corresponds to approximately 2652 optimization steps. During training, model checkpoints were periodically saved, and the checkpoint selected by early stopping was obtained at step 7500, corresponding approximately to epoch 3. This checkpoint was therefore used as the final model for evaluation on the test set.

## 4.2 Quantitative Evaluation

The performance of the base pretrained model and the fine-tuned model was evaluated using a combination of textual similarity metrics and protocol-specific structural metrics. Since protocol generation is not purely a free-text generation

task, the evaluation framework was designed to measure not only semantic similarity to the reference protocol, but also procedural structure, action consistency, parameter correctness, and output validity.

Table 4.1 summarizes the results across the complete set of evaluation metrics. Overall, the comparison shows that domain-specific fine-tuning substantially improves the model's ability to generate structured protocol outputs. The most pronounced gains are observed in the metrics that directly capture procedural structure, including Step Precision, Step Recall, Step F1, Action F1, and Structural Validity. For example, Structural Validity increases from 0.001308 in the base model to 0.835132 after fine-tuning, while Action F1 improves from 0.010249 to 0.382898. Similarly, step-level metrics such as Step Precision and Step Recall show large improvements, indicating that the fine-tuned model is significantly more capable of generating identifiable experimental steps and actions.

Table 4.1: Comparison between the base model and the fine-tuned model across evaluation metrics

| Metric | Base Model | Fine-tuned Model |
|---|---|---|
| EM | 0.000000 | 0.000000 |
| Token F1 | 0.214229 | 0.339913 |
| ROUGE-L F1 | 0.148915 | 0.302558 |
| BLEU-4 | 0.040508 | 0.143108 |
| BERTScore F1 | 0.804700 | 0.854300 |
| Step Precision | 0.001143 | 0.146588 |
| Step Recall | 0.002444 | 0.162672 |
| Step F1 | 0.001400 | 0.135339 |
| Step Match Avg. Sim. | 0.019935 | 0.511471 |
| Step Count Ratio | 0.489299 | 0.590439 |
| Step Order LCS | 0.013518 | 0.313600 |
| Action F1 | 0.010249 | 0.382898 |
| Parameter F1 | 0.044346 | 0.053878 |
| Structural Validity | 0.001308 | 0.835132 |
| Length Ratio | 2.910485 | 4.214468 |
| Distinct-2 | 0.607226 | 0.378683 |
| Distinct-3 | 0.752518 | 0.458434 |

All metric values are reported with six decimal precision to preserve numerical accuracy.

The evaluation results show that the improvements obtained through domain-specific fine-tuning are particularly evident in the metrics that measure procedural structure and action-level correctness. Metrics such as Step Precision, Step Recall,

Step F1, Action F1, and Structural Validity specifically evaluate whether the generated output follows the expected representation of laboratory protocols in terms of identifiable steps, actions, and structural format.

Among these metrics, Structural Validity exhibits the most substantial improvement, increasing from 0.001308 in the base model to 0.835132 in the fine-tuned model. This indicates that while the base model almost never produces outputs conforming to the required structured representation, the fine-tuned model generates structurally valid protocols in the majority of cases.

Substantial improvements are also observed in action-level and step-level metrics. Action F1 increases from 0.010249 to 0.382898, while Step F1 increases from 0.001400 to 0.135339. These changes indicate that the fine-tuned model is considerably more capable of generating recognizable experimental actions and organizing them into ordered procedural steps.

Taken together, these results suggest that the primary effect of fine-tuning is not merely improved textual similarity, but a significantly enhanced ability to reproduce the procedural structure required for machine-readable laboratory protocols.

Figure 4.2 presents the evaluation metrics that most directly reflect the model's ability to reproduce the procedural structure of laboratory protocols.

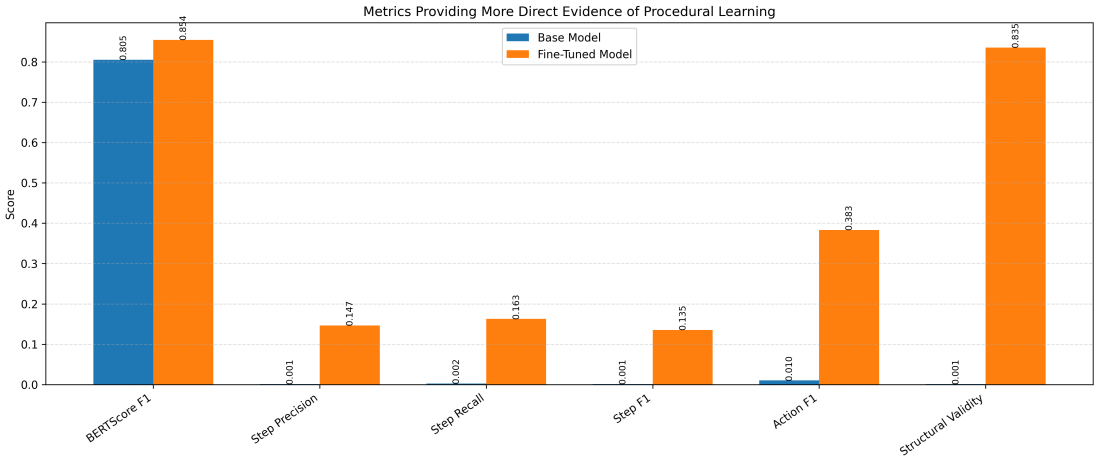


Figure 4.2: Comparison between the base model and the fine-tuned model on metrics that directly reflect procedural learning. These metrics evaluate the ability of the model to generate valid structured outputs, reproduce protocol steps, and preserve action-level procedural information.

These metrics are particularly important because they go beyond surface-level textual similarity and evaluate whether the generated output follows the expected procedural representation. In other words, they provide more direct evidence that

the model has learned the structure of laboratory workflows rather than only their general semantic content.

The results show substantial improvements across all direct procedural metrics after fine-tuning. Step precision increases from approximately 0.001 to 0.147, step recall rises from 0.002 to 0.163, and step F1 increases from approximately 0.001 to 0.135. Although the absolute values remain moderate, the magnitude of the improvement is large and indicates that the fine-tuned model is significantly more capable of generating identifiable procedural steps than the base model.

A similar pattern is observed for Action F1, which increases from 0.010 to 0.383. This suggests that fine-tuning substantially improves the model's ability to produce recognizable experimental actions, which is a core requirement for structured protocol generation.

The most striking result is obtained for structural validity, which increases from approximately 0.001 in the base model to 0.835 in the fine-tuned model. This shows that the base model almost never produces outputs in the expected structured format, whereas the fine-tuned model produces structurally valid protocols in the large majority of test examples.

Taken together, these results indicate that the main impact of fine-tuning is not merely to improve wording, but to enable the model to internalize the procedural form of the target protocol representation.

Figure 4.3 presents additional evaluation metrics that capture semantic similarity, lexical overlap, output length, ordering behavior, and diversity.

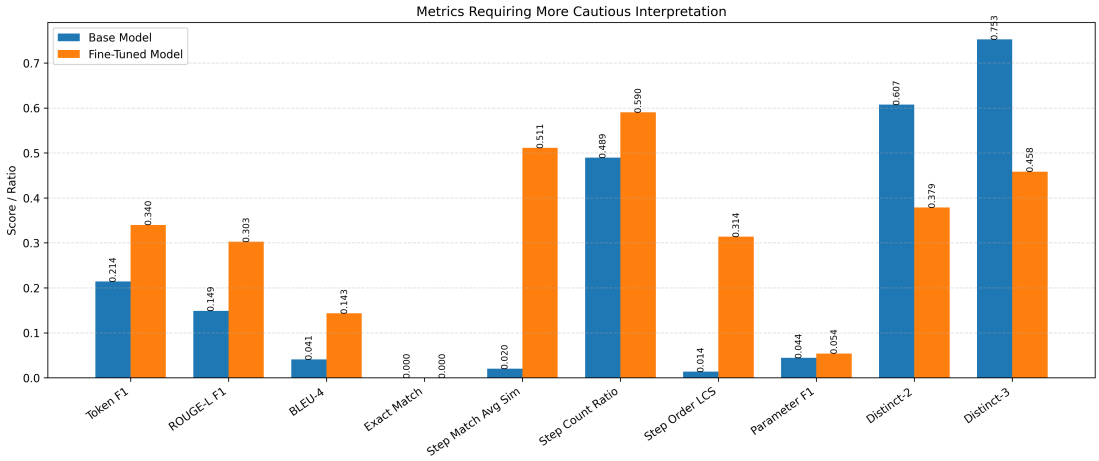


Figure 4.3: Comparison between the base model and the fine-tuned model on metrics that provide complementary but less direct evidence of procedural generation quality. These metrics include semantic similarity, lexical overlap, step alignment properties, and output diversity.

These metrics provide useful complementary information about the quality of the generated text, but they should be interpreted more carefully because they do not always correlate directly with procedural correctness. For example, lexical and semantic similarity metrics evaluate how closely the generated protocol resembles the reference text, but two protocols may express the same experimental procedure using different wording. Similarly, diversity and length-related metrics capture stylistic properties of the generated outputs rather than the correctness of the experimental workflow itself. For this reason, these metrics should be interpreted together with the procedural metrics discussed above.

Textual similarity metrics improve consistently after fine-tuning. BERTScore F1 increases from 0.8047 to 0.8543, ROUGE-L F1 increases from 0.149 to 0.303, and BLEU-4 increases from 0.041 to 0.143. These results indicate that the fine-tuned model generates outputs that are both semantically and lexically more aligned with the reference protocols.

Token F1 also increases from 0.214 to 0.340, showing improved overlap at the token level. Similarly, step match average similarity increases strongly from 0.020 to 0.511, suggesting that even when the generated steps are not exact matches, the fine-tuned model produces step content that is substantially closer to the reference procedures.

The step count ratio also improves from 0.489 to 0.590. This indicates that the fine-tuned model generates a number of steps that is closer to the reference protocols, although some mismatch in protocol length and granularity still remains. Step order LCS increases from 0.014 to 0.314, which further suggests that fine-tuning improves the ordering consistency of the generated procedural sequence.

Parameter F1 increases only modestly, from 0.044 to 0.054. This suggests that parameter generation remains more challenging than action generation, possibly because parameter expressions are more heterogeneous and may depend on fine-grained contextual details such as concentrations, durations, temperatures, or materials.

Some metrics, however, should not be interpreted in a purely monotonic way. In particular, Distinct-2 decreases from 0.607 to 0.379 and Distinct-3 decreases from 0.753 to 0.458. In open-ended creative text generation, such a decrease could be interpreted negatively because it reflects lower lexical diversity. In the present setting, however, lower diversity is not necessarily undesirable. Laboratory protocols are highly standardized procedural texts, and a good model is expected to reuse recurring domain-specific action patterns and conventional formulations. Therefore, the reduction in distinct n-gram ratios can also be interpreted as evidence that the fine-tuned model generates more regularized and task-consistent outputs.

A similar caution applies to the length-based metrics. The length ratio of the

fine-tuned model is higher than that of the base model. This does not automatically mean that the model is worse; rather, it suggests that the fine-tuned model tends to produce more complete and explicitly structured outputs. In procedural generation tasks, longer outputs may in fact reflect the inclusion of steps, actions, and parameters that are absent in shorter but less informative generations. For this reason, length-related metrics should always be interpreted together with structural validity and action-level performance rather than in isolation.

## 4.3 Qualitative Analysis

In addition to quantitative evaluation, qualitative analysis helps illustrate the behavioral differences between the base model and the fine-tuned model when responding to the same instruction.

Table 4.2 shows a shortened example for the prompt asking the model to generate a protocol for preparing alginate-based biomaterials for regenerative medicine applications.

Both the base model and the fine-tuned model were evaluated using exactly the same input prompt. Importantly, the prompt does not explicitly specify the structured function-based representation used in the reference protocols. Instead, the prompt simply requests the generation of a protocol for the given experimental task.

The qualitative example highlights a clear difference in generation behavior between the two models. The base pretrained model produces a plausible laboratory-style description written in natural language, but it does not follow the structured protocol representation expected by the dataset. As a result, the generated output lacks the explicit action-based structure used to represent experimental procedures.

In contrast, the fine-tuned model generates outputs that closely resemble the target protocol format. The generated steps follow the expected structured representation and include explicit procedural actions and parameters. Although the generated protocol is not an exact match to the reference protocol, it is structurally valid and substantially closer to the intended machine-readable representation.

This example illustrates the main effect of domain-specific fine-tuning: the model learns not only to generate plausible procedural text, but also to reproduce the structured representation required for computational protocol processing.

Table 4.2: Qualitative comparison between the base model, the fine-tuned model, and the reference protocol for an alginate-based biomaterial preparation task. Only representative excerpts are shown for readability.

**Instruction**

Please create a protocol to prepare alginate-based biomaterials for regenerative medicine applications.

**Reference Protocol (excerpt)**

```
# Title: Alginate-Based Biomaterial Preparation for
Regenerative Medicine
#1 prepare_standard_solution(solution_name='alginate solution',
concentration='1% w/v')
#2 dissolve_substance(substance='calcium chloride',
solvent='water', concentration='2% w/v')
#3 mix_components(component_1='alginate solution',
component_2='calcium chloride', ratio='1:1')
#4 incubate(duration='30 minutes', temperature='room
temperature')
```

**Base Model Output (excerpt)**

```
Protocol: Preparation of Alginate-Based Biomaterials for
Regenerative Medicine Applications
1. Preparation of Alginate Solution: Weigh 1 g of sodium
alginate and dissolve it in 100 mL of distilled water.
2. Preparation of Calcium Chloride Solution: Weigh 1 g of
calcium chloride and dissolve it in 100 mL of distilled water.
3. Formation of Alginate Hydrogel: Mix the alginate solution
with the calcium chloride solution in a 1:1 ratio.
```

**Fine-Tuned Model Output (excerpt)**

```
# Title: Preparation of Alginate-Based Biomaterials
#1 prepare_standard_solution(solution_name="alginate solution",
concentration="1% w/v", solvent="water")
#2 prepare_standard_solution(solution_name="calcium chloride
solution", concentration="100 mM", solvent="water")
#3 mix_components(component_1="alginate solution",
component_2="calcium chloride solution", ratio="1:1")
#4 incubate(time="30 minutes", temperature="room temperature")
```

# Chapter 5

# DISCUSSION

## 5.1 Interpretation of Results

An important observation from the evaluation results is the difference between improvements in traditional textual similarity metrics and improvements in protocol-specific procedural metrics. While metrics such as BLEU, ROUGE-L, Token F1, and BERTScore measure lexical and semantic similarity between generated and reference protocols, they do not directly capture whether the generated output follows the procedural structure of laboratory workflows. These observations are consistent with recent studies indicating that large language models show promising capabilities in biological reasoning and experimental knowledge synthesis, although significant challenges remain for complex procedural tasks [13, 14].

In contrast, protocol-specific metrics such as Step Precision, Step Recall, Action F1, and Structural Validity evaluate whether the generated protocols reproduce identifiable experimental steps, actions, and structured procedural representations. The experimental results show that improvements in these procedural metrics are substantially larger than those observed in purely textual similarity metrics.

This difference suggests that domain-specific fine-tuning primarily improves the model's ability to reproduce the structured procedural form of laboratory protocols rather than simply increasing textual similarity. Such behavior is particularly important for applications involving machine-readable protocols, where structural correctness and reproducibility are often more critical than superficial textual similarity.

The strongest result supporting this interpretation is the improvement in structural validity...

## 5.2 Impact of Dataset Normalization

An important component of the proposed framework is the normalization pipeline used to convert heterogeneous protocol descriptions into a unified structured representation.

Experimental protocols collected from different repositories often exhibit significant variability in writing style, formatting conventions, and level of detail. Such variability introduces noise into training data and makes it more difficult for language models to learn consistent procedural patterns.

The normalization pipeline developed in this work addresses this challenge by transforming raw protocol descriptions into a standardized representation that explicitly encodes experimental steps, actions, and parameters. By reducing linguistic variability and enforcing structural consistency, the normalization process allows the model to focus on procedural relationships rather than textual inconsistencies.

The improvements observed in structural evaluation metrics suggest that this normalization step plays an important role in enabling the model to learn more reliable representations of experimental workflows. In particular, the large improvement in structural validity indicates that the model successfully learns the expected protocol schema when trained on normalized structured data.

## 5.3 Effectiveness of Structured Protocol Representations

Another key aspect of the proposed approach is the use of structured protocol representations for both training and evaluation.

Traditional protocol descriptions are typically written as free-form natural language instructions. While such descriptions are convenient for human readers, they often contain implicit assumptions, missing parameters, or ambiguous step boundaries that complicate computational processing.

By converting protocols into structured representations that explicitly encode experimental actions and parameters, it becomes possible to represent laboratory workflows in a machine-interpretable form. This structured representation serves as an intermediate format between natural language descriptions and computational systems that may analyze or execute experimental procedures.

The experimental results suggest that large language models can successfully learn such representations when trained on sufficiently consistent datasets. In particular, the improvements observed in step-level and action-level metrics indicate that the model learns to reproduce the sequential organization of experimental procedures.

This observation supports the idea that structured representations can facilitate the application of language models to scientific workflows where procedural correctness and reproducibility are critical.

## 5.4 Limitations of the Proposed Approach

Despite the promising results obtained in this work, several limitations should be acknowledged.

First, the dataset used for training and evaluation consists of protocols collected from publicly available online repositories. While these sources provide valuable information about experimental workflows, they may vary in quality, completeness, and level of detail. Some protocols may omit parameters or rely on implicit domain knowledge, which can introduce noise into the training data.

Second, the evaluation framework relies primarily on automatic metrics that measure textual similarity and structural consistency. Although these metrics provide useful indicators of model performance, they cannot fully capture whether generated protocols are experimentally valid or practically executable in laboratory environments.

Third, although fine-tuning substantially improves the structural consistency of generated outputs, the model may still occasionally generate incomplete steps, redundant actions, or minor deviations from the expected protocol format. This limitation reflects the inherent difficulty of modeling complex procedural knowledge using language models.

Finally, the experiments conducted in this work rely on a single model architecture and a specific fine-tuning configuration. While parameter-efficient fine-tuning enables training large models with limited computational resources, it may restrict exploration of alternative model architectures or training strategies.

## 5.5 Implications for Biofabrication and Scientific Automation

The results of this work suggest that combining structured protocol datasets with large language models may provide a promising foundation for future tools supporting experimental research.

In domains such as biofabrication, experimental procedures are often complex and involve multiple steps, materials, and environmental conditions. Automated or assisted protocol generation systems could help researchers document experiments more consistently, explore alternative experimental workflows, and improve the reproducibility of laboratory procedures.

More broadly, structured representations of laboratory protocols could contribute to the development of digital laboratory infrastructures in which experimental workflows are stored, analyzed, and potentially executed in automated environments.

While the present work focuses primarily on protocol representation and generation, it highlights the potential role of language models as components of future intelligent laboratory systems capable of supporting experimental design, documentation, and analysis.

# Chapter 6

# CONCLUSION

This thesis investigated the problem of representing and generating biofabrication protocols using large language models and structured procedural representations. The motivation for this work arises from challenges in reproducibility, protocol standardization, and the increasing need for computational tools capable of supporting experimental research workflows.

Experimental protocols are often written in natural language and exhibit substantial variability in formatting, terminology, and level of detail. Such heterogeneity makes it difficult for computational systems to interpret procedural information or automatically process experimental workflows. In this work, we addressed this challenge by combining dataset normalization techniques with domain-adapted large language models in order to generate structured and machine-readable protocol representations.

## 6.1 Summary of Contributions

The first contribution of this thesis is the construction of a dataset of biofabrication protocols collected from multiple online repositories and scientific sources. The collected protocols were retrieved using automated queries and processed using a data collection pipeline designed to gather heterogeneous protocol descriptions from different repositories.

The second contribution is the design and implementation of a normalization pipeline that converts raw protocol descriptions into a unified structured representation. This pipeline extracts procedural information such as experimental steps, actions, and parameters, enabling the transformation of heterogeneous textual descriptions into a consistent format suitable for machine learning.

The third contribution is the application of large language models to the task of structured protocol generation. Using the normalized dataset, a transformer-based language model was fine-tuned using parameter-efficient techniques in order

to adapt the model to the biofabrication domain.

A further contribution of this work is the development of an evaluation framework that combines traditional natural language generation metrics with protocol-specific structural metrics. This framework enables a more comprehensive assessment of generated protocols by measuring both textual similarity and procedural correctness.

## 6.2 Key Findings

The experimental evaluation confirms that domain-specific fine-tuning substantially improves the generation of structured laboratory protocols. Compared to the base pretrained model, the fine-tuned model produces outputs that are significantly more consistent with the expected structured representation of experimental workflows.

In particular, the structural validity of generated protocols increases dramatically after fine-tuning, rising from 0.001308 in the base model to 0.835132 in the fine-tuned model. This result indicates that while the base model rarely produces outputs conforming to the structured protocol representation, the fine-tuned model generates structurally valid protocols in the large majority of cases.

Large improvements are also observed in step-level and action-level metrics. For example, Action F1 increases from 0.010249 to 0.382898, Step Precision from 0.001143 to 0.146588, and Step Recall from 0.002444 to 0.162672. These results suggest that the model learns to reproduce key procedural elements such as experimental actions, ordered steps, and associated parameters.

Textual similarity metrics also improve after fine-tuning, indicating that the generated protocols become both semantically and lexically closer to the reference procedures. However, the improvements observed in procedural metrics are substantially larger, suggesting that the main effect of fine-tuning is the acquisition of structured procedural representations rather than only improved textual similarity.

Overall, the results highlight the importance of combining domain-specific datasets with structured representations when applying language models to procedural scientific text generation.

## 6.3 Limitations

Despite the promising results obtained in this work, several limitations remain.

First, the dataset used in this study is limited to protocols collected from publicly available repositories. Although these sources provide valuable information

about experimental workflows, they may contain inconsistencies, missing parameters, or varying levels of detail.

Second, the evaluation framework relies primarily on automatic evaluation metrics. While these metrics provide useful indicators of model performance, they cannot fully determine whether generated protocols are experimentally valid or practically executable in real laboratory environments.

Third, although the fine-tuned model produces structurally valid outputs in many cases, it may still generate incomplete steps, redundant instructions, or minor deviations from the expected protocol schema.

Finally, the experiments conducted in this work focus on a single model architecture and fine-tuning configuration. Exploring alternative architectures, model sizes, or training strategies could potentially lead to further improvements.

## 6.4 Future Work

Several directions for future research emerge from this work.

First, the dataset could be expanded to include protocols from additional experimental domains such as molecular biology, tissue engineering, and synthetic biology. A larger and more diverse dataset would enable more robust training and evaluation of language models for procedural text generation.

Second, future work could investigate the integration of multimodal information such as experimental images, diagrams, or laboratory metadata. Such information could provide additional contextual signals that may improve the quality of generated protocols.

Third, structured protocol representations could potentially be integrated with laboratory automation systems or robotic platforms. This integration could enable partially automated execution of experimental workflows. Recent research directions also explore the integration of language models with automated scientific workflows and robotic laboratory platforms [37, 39].

Another promising direction is the development of validation mechanisms for generated protocols. Combining language models with rule-based verification systems or domain knowledge constraints could help detect inconsistencies or unsafe experimental instructions before protocols are used in laboratory settings.

Finally, future research could explore more advanced language model architectures capable of explicitly modeling procedural dependencies between experimental steps.

## 6.5 Concluding Remarks

In conclusion, this thesis demonstrates that combining structured protocol representations with domain-adapted large language models provides a promising approach for improving the representation and generation of experimental workflows.

By transforming heterogeneous protocol descriptions into structured datasets and adapting language models to this domain, it becomes possible to generate outputs that more closely resemble machine-readable laboratory protocols. Such approaches may contribute to improving reproducibility, accessibility, and automation in experimental science.

As language models continue to evolve, their integration with structured scientific data may play an important role in the development of future intelligent laboratory systems capable of supporting experimental design, documentation, and analysis.

The results of this thesis suggest that the main benefit of domain adaptation is not only improved textual generation, but the ability to reproduce structured procedural representations required for computational protocol processing.

# Bibliography


[1] J. Groll, T. Boland, T. Blunk, J. A. Burdick, D.-W. Cho, P. D. Dalton, B. Derby, G. Forgacs, Q. Li, V. A. Mironov, L. Moroni, M. Nakamura, W. Shu, S. Takeuchi, G. Vozzi, T. B. F. Woodfield, T. Xu, J. J. Yoo, and J. Malda, "Biofabrication: reappraising the definition of an evolving field," vol. 8, no. 1, p. 013001. [Online]. Available: https://iopscience.iop.org/article/10.1088/1758-5090/8/1/013001

[2] L. Moroni, T. Boland, J. A. Burdick, C. De Maria, B. Derby, G. Forgacs, J. Groll, Q. Li, J. Malda, V. A. Mironov, C. Mota, M. Nakamura, W. Shu, S. Takeuchi, T. B. Woodfield, T. Xu, J. J. Yoo, and G. Vozzi, "Biofabrication: A guide to technology and terminology," vol. 36, no. 4, pp. 384–402. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167779917302792

[3] M. Baker, "1,500 scientists lift the lid on reproducibility," vol. 533, no. 7604, pp. 452–454. [Online]. Available: https://www.nature.com/articles/533452a

[4] B. Bartley, J. Beal, M. Rogers, D. Bryce, R. P. Goldman, B. Keller, P. Lee, V. Biggers, J. Nowak, and M. Weston, "Building an open representation for biological protocols," pages: 2022.07.05.498808 Section: New Results. [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.07.05.498808v1

[5] I. Ivanov, "BioLP-bench: Measuring understanding of biological lab protocols by large language models," pages: 2024.08.21.608694 Section: New Results. [Online]. Available: https://www.biorxiv.org/content/10.1101/2024.08.21.608694v4

[6] Y. Liu, L. Lv, X. Zhang, L. Yuan, and Y. Tian, "BioProBench: Comprehensive dataset and benchmark in biological protocol understanding and reasoning." [Online]. Available: http://arxiv.org/abs/2505.07889

[7] R. Bardini and S. D. Carlo, "Computational methods for biofabrication in tissue engineering and regenerative medicine - a literature review," vol. 23, pp. 601–616. [Online]. Available: https://www.csbj.org/article/S2001-0370(23)00511-1/fulltext

[8] V. Ananthanarayanan and W. Thies, "Biocoder: A programming language for standardizing and automating biology protocols," vol. 4, no. 1, p. 13. [Online]. Available: https://doi.org/10.1186/1754-1611-4-13

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need." [Online]. Available: http://arxiv.org/abs/1706.03762

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding." [Online]. Available: http://arxiv.org/abs/1810.04805

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners." [Online]. Available: http://arxiv.org/abs/2005.14165

[12] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models." [Online]. Available: http://arxiv.org/abs/2302.13971

[13] W. Ruan, Y. Lyu, J. Zhang, J. Cai, P. Shu, Y. Ge, Y. Lu, S. Gao, Y. Wang, P. Wang, L. Zhao, T. Wang, Y. Liu, L. Fang, Z. Liu, Z. Liu, Y. Li, Z. Wu, J. Chen, H. Jiang, Y. Pan, Z. Yang, J. Chen, S. Liang, W. Zhang, T. Ma, Y. Dou, J. Zhang, X. Gong, Q. Gan, Y. Zou, Z. Chen, Y. Qian, S. Yu, J. Lu, K. Song, X. Wang, A. Sikora, G. Li, X. Li, Q. Li, Y. Wang, L. Zhang, Y. Abate, L. He, W. Zhong, R. Liu, C. Huang, W. Liu, Y. Shen, P. Ma, H. Zhu, Y. Yan, D. Zhu, and T. Liu, "Large language models for bioinformatics." [Online]. Available: http://arxiv.org/abs/2501.06271

[14] Z. Lu, Y. Peng, T. Cohen, M. Ghassemi, C. Weng, and S. Tian, "Large language models in biomedicine and health: current research landscape and future directions," vol. 31, no. 9, pp. 1801–1811. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11339542/

[15] W. Wang, L. Gu, L. Zhang, Y. Luo, Y. Dai, C. Shen, L. Xie, B. Lin, X. He, and J. Ye, "SciPIP: An LLM-based scientific paper idea proposer," version: 1. [Online]. Available: http://arxiv.org/abs/2410.23166

[16] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, "BioGPT: Generative pre-trained transformer for biomedical text generation and mining," vol. 23, no. 6, p. bbac409. [Online]. Available: http://arxiv.org/abs/2210.10341

[17] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science." [Online]. Available: http://arxiv.org/abs/2211.09085

[18] Q. Zhang, K. Ding, T. Lyv, X. Wang, Q. Yin, Y. Zhang, J. Yu, Y. Wang, X. Li, Z. Xiang, K. Feng, X. Zhuang, Z. Wang, M. Qin, M. Zhang, J. Zhang, J. Cui, T. Huang, P. Yan, R. Xu, H. Chen, X. Li, X. Fan, H. Xing, and H. Chen, "Scientific large language models: A survey on biological & chemical domains." [Online]. Available: http://arxiv.org/abs/2401.14656

[19] O. O'Donoghue, A. Shtedritski, J. Ginger, R. Abboud, A. E. Ghareeb, J. Booth, and S. G. Rodriques, "BioPlanner: Automatic evaluation of LLMs on protocol planning in biology." [Online]. Available: http://arxiv.org/abs/2310.10632

[20] S. Jiang, D. Evans-Yamamoto, D. Bersenev, S. K. Palaniappan, and A. Yachie-Kinoshita, "ProtoCode: Leveraging large language models (LLMs) for automated generation of machine-readable PCR protocols from scientific publications," vol. 29, no. 3, p. 100134. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2472630324000165

[21] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs." [Online]. Available: http://arxiv.org/abs/2305.14314

[22] S. V. Murphy and A. Atala, "3d bioprinting of tissues and organs," vol. 32, no. 8, pp. 773–785.

[23] C. Mandrycky, Z. Wang, K. Kim, and D.-H. Kim, "3d bioprinting for engineering complex tissues," vol. 34, no. 4, pp. 422–434. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0734975015300665

[24] "Protocols.io," https://www.protocols.io, accessed: April 2025.

[25] "Openwetware," https://openwetware.org, accessed: April 2025.

[26] A.-M. Anhel, L. Alejaldre, and A. Goni-Moreno, "The laboratory automation protocol (LAP) format and repository: A platform for enhancing workflow efficiency in synthetic biology," vol. 12, no. 12, pp. 3514–3520. [Online]. Available: https://doi.org/10.1021/acssynbio.3c00397

[27] R. Tamari, F. Bai, A. Ritter, and G. Stanovsky, "Process-level representation of scientific protocols with interactive annotation." [Online]. Available: http://arxiv.org/abs/2101.10244

[28] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models." [Online]. Available: http://arxiv.org/abs/2001.08361

[29] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal large language models." [Online]. Available: http://arxiv.org/abs/2203.15556

[30] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models." [Online]. Available: http://arxiv.org/abs/2106.09685

[31] T. Zheng, Z. Deng, H. T. Tsang, W. Wang, J. Bai, Z. Wang, and Y. Song, "From automation to autonomy: A survey on large language models in scientific discovery." [Online]. Available: http://arxiv.org/abs/2505.13259

[32] O. A. Sarumi and D. Heider, "Large language models and their applications in bioinformatics," vol. 23, pp. 3498–3505. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2001037024003209

[33] T. Yang, Y. Xiao, Z. Bao, J. Hao, and J. Peng, "The rise and potential opportunities of large language model agents in bioinformatics and biomedicine," vol. 26, no. 6, p. bbaf601. [Online]. Available: https://doi.org/10.1093/bib/bbaf601

[34] C. Kulkarni, W. Xu, A. Ritter, and R. Machiraju, "An annotated corpus for machine reading of instructions in wet lab protocols." [Online]. Available: http://arxiv.org/abs/1805.00195

[35] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. v. Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab,

P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, "On the opportunities and risks of foundation models." [Online]. Available: http://arxiv.org/abs/2108.07258

[36] S. Yi, J. Lim, and J. Yoon, "ProtoMed-LLM: An automatic evaluation framework for large language models in medical protocol formulation." [Online]. Available: http://arxiv.org/abs/2410.04601

[37] Y. Luo, L. Shi, Y. Li, A. Zhuang, Y. Gong, L. Liu, and C. Lin, "From intention to implementation: Automating biomedical research via LLMs," vol. 68, no. 7, p. 170105. [Online]. Available: http://arxiv.org/abs/2412.09429

[38] T. Inagaki, A. Kato, K. Takahashi, H. Ozaki, and G. N. Kanda, "LLMs can generate robotic scripts from goal-oriented instructions in biological laboratory automation." [Online]. Available: http://arxiv.org/abs/2304.10267

[39] K. Darvish, M. Skreta, Y. Zhao, N. Yoshikawa, S. Som, M. Bogdanovic, Y. Cao, H. Hao, H. Xu, A. Aspuru-Guzik, A. Garg, and F. Shkurti, "ORGANA: A robotic assistant for automated chemistry experimentation and characterization." [Online]. Available: http://arxiv.org/abs/2401.06949

[40] X. Xu and R. Sankar, "Large language model agents for biomedicine: A comprehensive review of methods, evaluations, challenges, and future directions," vol. 16, no. 10, p. 894. [Online]. Available: https://www.mdpi.com/2078-2489/16/10/894

[41] T. Hartung, "AI, agentic models and lab automation for scientific discovery — the beginning of scAInce," vol. 8, p. 1649155. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC12426084/

[42] M. P. Abrate, R. Smeriglio, R. Bardini, A. Savino, and S. D. Carlo, "Fast and accurate LSTM meta-modeling of TNF-induced tumor resistance in vitro," pages: 2024.08.12.607535 Section: New Results. [Online]. Available: https://www.biorxiv.org/content/10.1101/2024.08.12.607535v1

[43] "Openalex," https://openalex.org, accessed: April 2025.

[44] "Pubmed," https://pubmed.ncbi.nlm.nih.gov, accessed: April 2025.

[45] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. v. d. Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. v. d. Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. d. Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand,

A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury,

S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, "The llama 3 herd of models." [Online]. Available: http://arxiv.org/abs/2407.21783

[46] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: http://arxiv.org/abs/1406.1078