GOFinder-AI: Rapid and Explainable Gene Ontology Term Assignment Using Large Language Models

Aws Almir Ahmad

Thesis submitted to the University of Ottawa

In partial fulfillment of the requirements for the

MSc degree in Biochemistry specializing in Bioinformatics

Department of Biochemistry, Microbiology and Immunology

Faculty of Medicine

University of Ottawa

To my mother,

Dr. Wafa Hadid.

Acknowledgments

Having joined Dr. Arvind Mer's lab as an undergraduate student with limited bioinformatics background, I can only describe my time under his supervision as a rich learning experience, both academically and personally. In the lab, Professor Arvind taught me to become an early-career bioinformatician in the emerging world of large language models (LLMs) and in the fields of text mining and machine learning. Personally, he was more than just a supervisor. He was a mentor who gave me time and guidance as I transitioned to this new phase of my academic career. As part of this and the other projects I worked on in the lab, Professor Arvind connected me with numerous software engineers and computer science students, with whom I worked to advance GOFinder-AI and from whom I learned a great deal. Special shoutouts to Mostafa Othman, Jordan Lau, and the many other amazing students who were part of developing this tool at every step of the way.

My experience as a Master's student was further enriched by my thesis advisory committee members, Professor Mathieu Lavallée-Adam and Professor Xiaojian Shao, both of whom, along with Professor Arvind, have been unconditionally supportive, providing me with the guidance I needed to become a more confident scientist and bioinformatician. I owe them my deepest appreciation and gratitude.

My appreciation also extends to the members of my lab, with special shoutouts to Ben Patrick, Quin Brohart, Amir Ebrahimi, and Stefan Wallin for their support and, above all, our friendship, which I am honoured to carry forward.

Much of my gratitude also goes to my god, before all, without whom I have none, and to my mother, Dr. Wafa Hadid, my hero in this life and my biggest supporter.

Abstract

Gene Ontology (GO) provides a structured vocabulary for describing the function of gene products. However, the rapid growth of biomedical literature makes manual GO curation increasingly difficult to sustain. Here, we present GOFinder-AI, a computational framework that supports literature-grounded GO annotation through pre-query text mining and large language model (LLM) inference. Given a biomedical text, the system identifies candidate GO annotations and produces supporting citations, explanatory reasoning, and linked biological entities. To improve task-specific performance, we fine-tuned multiple general-purpose LLMs (Llama-3.1-8B and Qwen3-8B) on a large, annotated dataset with more than 23,000 examples. Model performance was assessed using grouped 4-fold cross-validation, followed by evaluation on an independent test set containing >7000 gene-GO associations. Fine-tuning markedly improved performance compared to zero-shot prompting. The fine-tuned Qwen3-8B-based system reported higher predictive accuracy than GPT-5 mini, Llama-3.1-8B, and its own zero-shot counterpart. Overall, when tested on over 3,500 annotations, GOFinder-AI achieved a cumulative accuracy of 95.32%. It completed document-level GO curation in under one minute on average. GOFinder-AI offers a scalable, interpretable, and transparent approach to automated GO curation.

Table of Contents

List of abbreviations

GO	Gene Ontology
BP	Biological Process
CC	Cellular Component
MF	Molecular Function
DAG	Directed Acyclic Graph
CAFA	Critical Assessment of Functional Annotation
RNA	Ribonucleic Acid
TF	Term Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
BM25	Best Matching 25
NLP	Natural Language Processing
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
LLM	Large Language Model
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
LoRA	Low-Rank Adaptation
NCBI	National Center for Biotechnology Information
PMID	PubMed Identifier [used in this thesis to refer to the documents sourced from PubMed]
PMC	PubMed Central
IEA	Inferred from Electronic Annotation
TAS	Traceable Author Statement
IDA	Inferred from Direct Assay

IMP	Inferred from Mutant Phenotype
ISS	Inferred from Sequence Similarity
IBA	Inferred from Biological aspect of Ancestor
ISA	Inferred from Sequence Alignment
ISO	Inferred from Sequence Orthology
ISM	Inferred from Sequence Model
RCA	Reviewed Computational Analysis
IPI	Inferred from Physical Interaction
IGI	Inferred from Genetic Interaction
EXP	Inferred from Experiment
IEP	Inferred from Expression Pattern
NAS	Non-traceable Author Statement
IC	Inferred by Curator
ND	No biological Data available
PAINT	Phylogenetic Annotation INference Tool
EC	Enzyme Commission
KEGG	Kyoto Encyclopedia of Genes and Genomes
SFT	Supervised Fine-tuning
RAG	Retrieval-Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
DPO	Direct Preference Optimization
TRL	Transformer Reinforcement Learning (HuggingFace library)
PDF	Portable Document Format
API	Application programming interface
GPU	Graphics Processing Unit
OA	Open Access
IQR	Interquartile Range

List of Figures

List of Equations

List of Tables

1. Introduction

1.1 Biological Entities

Biological entities, including genes, proteins, transcripts, isoforms, and other molecular products, are the functional units through which cellular processes are carried out and regulated. Accurate functional characterization of these entities is essential for molecular biology, functional genomics, and precision medicine, because biological interpretation ultimately depends on knowing what a given entity does, where it acts, and which processes it participates in. As high-throughput sequencing and molecular profiling continue to expand the number of characterized entities, the need for scalable, evidence-based functional annotation frameworks has become increasingly important [1].

1.2 Gene Ontology (GO) and Current Gaps in Functional Annotation

Gene Ontology (GO) is the most widely used framework for representing gene product function in a structured, computable manner. GO is a controlled vocabulary, a fixed set of terms with stable identifiers and explicit definitions, used to ensure that the same biological concept is described consistently across databases, organisms, and curators [2,3]. By providing this vocabulary, GO ensures that an annotation written today by one curator can be compared, queried, and reasoned over by software tools years later and across species. GO organizes biological knowledge into three coordinated namespaces, also called domains**:** Biological Process (BP), which describes the broader pathway or program — such as apoptotic process or DNA repair **—** in

which the gene product's activity contributes; Cellular Component (CC), which specifies where the gene product acts — for example, the nucleolus, the mitochondrial inner membrane, or a macromolecular complex such as the spliceosome**;** and Molecular Function (MF), which describes the biochemical activity of a gene product at the elemental level, such as protein kinase activity or ATP binding [2,3]. Within each domain, GO is organized as a hierarchical ontology in which broad parent terms are linked to more specific child terms through defined relationships such as "*is_a*" and "*part_of*". This structure allows functional knowledge to be represented at different levels of specificity while preserving biologically meaningful relationships between terms. Because GO provides a controlled vocabulary and hierarchical structure, it has become central to enrichment analysis, comparative genomics, pathway interpretation, and machine-learning pipelines that use its functional features as model inputs [4,5].

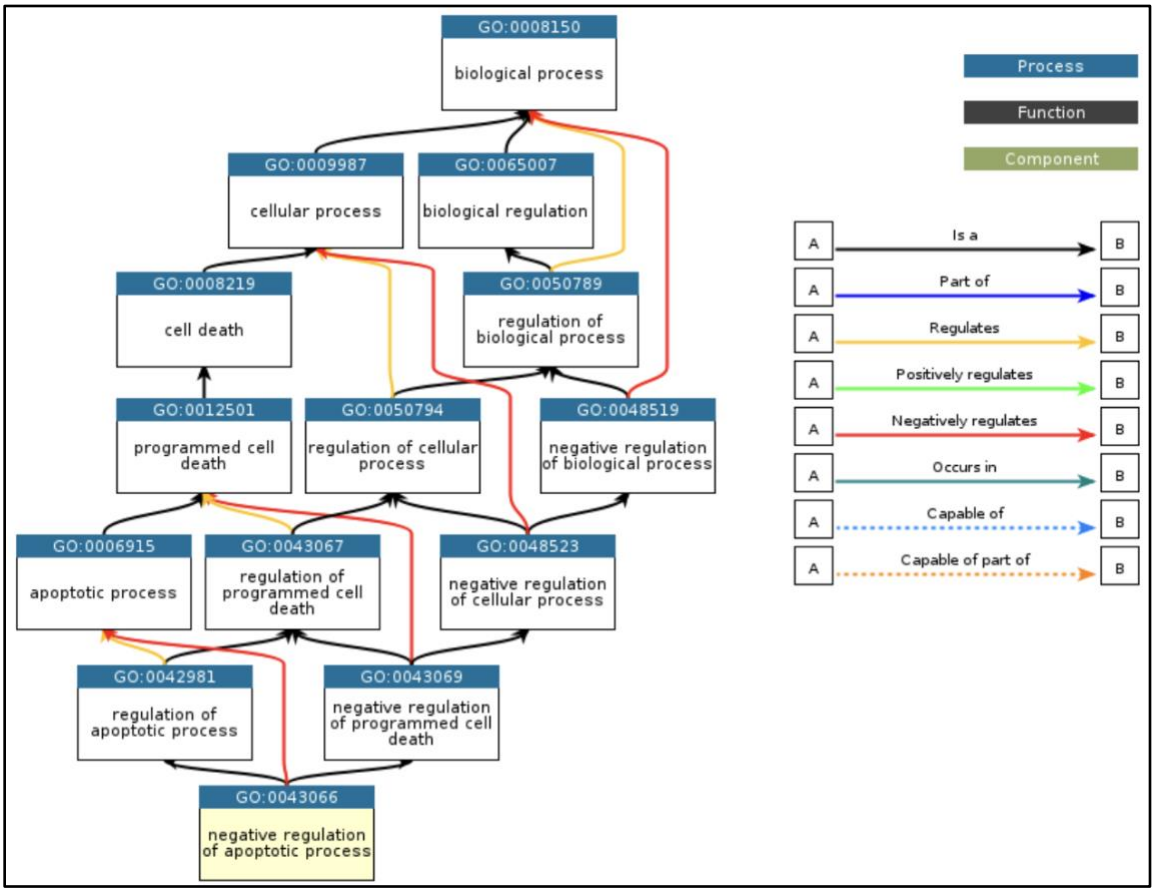

Figure 1. Hierarchical Structure of GO Terms. The figure shows a directed acyclic graph (DAG) illustrating the hierarchical organization of a single GO term within the Biological Process domain. Broad parent terms, such as biological process, branch into increasingly specific child terms via defined relationships, including *is_a*, *part_of*, regulates, positively regulates, and negatively regulates. This structure enables GO to represent biological knowledge at multiple levels of specificity and supports the computational interpretation of functional relationships among related terms [2,3].

Each gene–GO association is supported by an evidence code that records how the assignment was made. As shown in Figure 2 below, evidence codes fall into three broad classes: experimental codes such as IDA (Inferred from Direct Assay), IMP (Inferred from Mutant Phenotype), and IPI (Inferred from Physical Interaction), which are based on results reported in the primary literature; author-statement codes such as TAS (Traceable Author Statement), in which the annotation is supported by an explicit claim in a cited publication; and computational codes such as ISS (Inferred from Sequence or Structural Similarity), IBA (Inferred from Biological aspect of Ancestor), and IEA (Inferred from Electronic Annotation), in which the assignment is propagated from prior knowledge rather than read directly from new experimental evidence [3,6,7]. The GO ecosystem therefore relies on two complementary processes: manual curation, in which experienced biocurators read primary publications and assign evidence-linked annotations, and computational propagation, in which manually curated annotations are extended at scale to homologous, orthologous, or related gene products through sequence similarity, protein-family signatures, phylogenetic inference, or rule-based pipelines [3,6]. Manual curation provides the high-confidence backbone of GO but covers only a small fraction of the gene products of interest, while computational propagation provides scale but ultimately depends on the quality, specificity, and coverage of the manually curated assignments from which it is derived. The interplay between these two processes shapes both the strengths and the current gaps of GO.

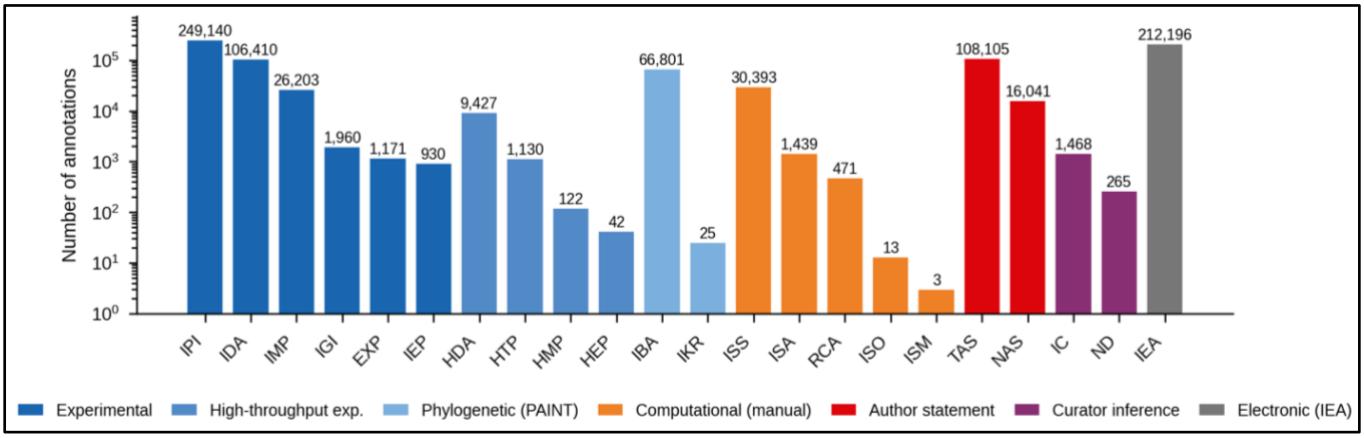

Figure 2. Evidence-Code Distribution across Human GO Annotations. Bar plot showing the number of GO annotations assigned to human gene products, broken down by GO evidence code (x-axis, log-scaled y-axis). Numerical labels above each bar indicate the absolute count for each code. Bars are colour-coded by curation modality. Experimental (dark blue) annotations are derived from primary experimental evidence and include Inferred from Physical Interaction (IPI), Inferred from Direct Assay (IDA), Inferred from Mutant Phenotype (IMP), Inferred from Genetic Interaction (IGI), Inferred from Experiment (EXP), and Inferred from Expression Pattern (IEP). High-throughput experimental annotations (medium blue) are the high-throughput counterparts of these codes (HDA, HTP, HMP, HEP). Phylogenetic (PAINT) annotations (light blue) are propagated through curated protein-family phylogenies and include Inferred from Biological aspect of Ancestor (IBA) and Inferred from Key Residues (IKR). Computational (manual) annotations (orange) are computationally inferred but reviewed by curators and include Inferred from Sequence or Structural Similarity (ISS), Inferred from Sequence Alignment (ISA), Reviewed Computational Analysis (RCA), Inferred from Sequence Orthology (ISO), and Inferred from Sequence Model (ISM). Author statement annotations (red) are based on assertions in the published literature: Traceable Author Statement (TAS) and Non-traceable Author Statement (NAS). Curator inference annotations (purple) include Inferred by Curator (IC) and No biological Data available (ND). Electronic (IEA) annotations (grey) are fully computational, unreviewed assignments. Data were obtained from the human GO Annotation File (taxon ID 9606) released by the GO Consortium [3].

1.3 NCBI-PubMed and the Literature Curation Bottleneck

Despite its importance, GO remains incomplete (see Figure 3). A major reason for the gaps in GO is the sheer scale of the biomedical literature. PubMed, a database developed and maintained by the National Center for Biotechnology Information (NCBI) at the U.S. National Library of Medicine, is the principal public repository of biomedical references and abstracts; it indexes over 40 million citations and serves as the main entry point for curators to identify

candidate evidence for GO annotations [8,9]. As shown in Figure 4, PubMed continues to expand at a pace that far exceeds what domain experts can manually process, with recent estimates indicating roughly 3,000 new biomedical articles per day [8,10]. In practice, this means that large amounts of experimentally relevant functional evidence remain embedded in text rather than being converted into structured annotations. For resources such as GO, the challenge is therefore not only how to represent knowledge but also how to identify, retrieve, and formalize it efficiently from an ever-growing body of literature [8,10].

This growth in the literature creates a curation bottleneck. Even when relevant evidence exists, curators must still locate the correct paper, identify the appropriate experimental signal, determine the most suitable GO term, and encode the annotation in a consistent, evidence-linked format. That manual workflow is reliable but labour-intensive and does not scale to current publication rates [3,8,10]. The result is a persistent lag between published biological findings and their incorporation into computable databases. For poorly annotated genes, this lag can be substantial. For entities studied across multiple publications or under multiple aliases, the burden is even greater. These conditions motivate the development of systems that can assist curators by narrowing the search space and surfacing high-probability, evidence-grounded annotation candidates.

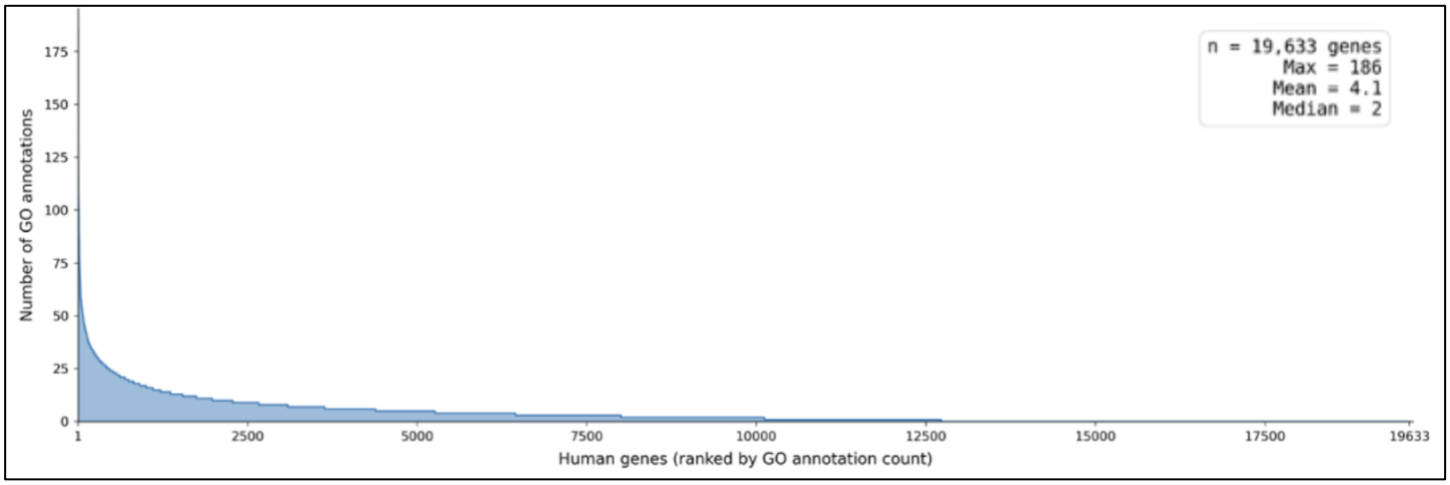

Figure 3. Distribution of GO annotations across Human Genes. Distribution of the number of GO annotations per human protein-coding gene (n = 19,633 genes). Genes are ranked along the x-axis in descending order of annotation count, and the y-axis shows the number of GO annotations assigned to each gene across all three GO domains (BP, CC, MF) and all evidence codes pooled. The distribution is strongly right-skewed: a small subset of intensively studied genes carries up to 186 annotations, whereas most genes carry only a handful (mean = 4.1, median = 2). Beyond approximately the top 5,000 ranked genes, annotation counts drop sharply and remain low across the remaining ~14,000 poorly annotated genes. Data were obtained from the same human GO Annotation File release as in Figure 2 [3].

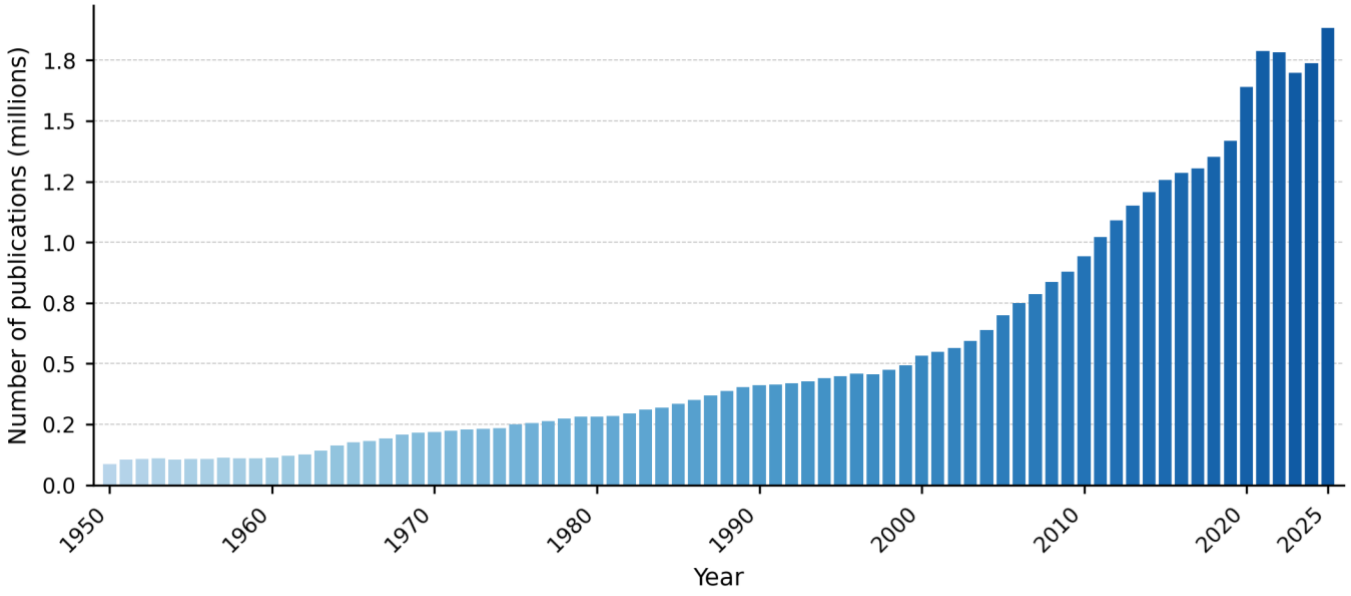

Figure 4. PubMed Publications per Year. Bar plot showing the annual number of publications indexed in PubMed from 1950 to 2026, based on NCBI-PubMed query results [9].

1.4 Existing Automated Curation Approaches and Their Limitations

Automated and semi-automated annotation methods already play an important role in the GO ecosystem [3,6,11,12,13]. A major category comprises annotations supported by the IEA evidence code [3,6,11]. These annotations are generated computationally, typically using methods based on sequence similarity, conserved protein domains, protein family membership, keyword mappings, enzyme classifications, or related rule-based pipelines. Their main strength is scale: they can annotate large numbers of gene products far more rapidly than literature-based manual curation [3,6,11].

A well-known example of IEA methods is InterPro2GO, which maps curated protein family or domain signatures from InterPro to GO terms and propagates these mappings to matching proteins [6,12]. This approach is highly effective for broad functional coverage, especially when a conserved domain strongly implies a recurring molecular activity. However, its limitations are also clear. The specificity of the resulting GO annotation depends on the granularity of the underlying InterPro signature, and domain-based mappings may remain broad when the same domain is associated with multiple biological contexts or functions [12,14]. As a result, InterPro2GO is well suited for scalable functional transfer but less able to capture condition-specific or experimentally nuanced claims described in individual publications.

A second approach is rule-based automatic annotation, such as UniProt's UniRule framework, in which expert-derived rules are applied when predefined sequence, taxonomic, or protein-family criteria are met [15,16]. Compared with naive homology transfer, rule-based systems can improve consistency and biological plausibility by encoding expert knowledge and biological constraints directly into the annotation process. However, they still depend on pre-existing rule coverage and

are inherently limited by what has already been formalized in the rule set. Consequently, they are powerful for recurring, well-characterized functional patterns but less suited to identifying newly published, literature-grounded functional evidence that has not yet been incorporated into existing annotation rules [15,16].

A third relevant category is phylogeny-informed propagation, particularly PAINT or PAN-GO-style annotation [3,13]. In these systems, curators infer function in an evolutionary context by reviewing experimentally supported annotations across a protein family and propagating selected functions along a phylogenetic tree. This approach is more biologically informed than purely automatic transfer and can yield higher-quality inferences because it explicitly incorporates evolutionary relationships and curator judgment [13]. However, it still does not replace direct article-level extraction of evidence from newly published literature. Its quality depends on existing family coverage, the available experimental base, and curator-guided interpretation of evolutionary conservation and divergence [13].

More broadly, evaluations of computational GO annotation have shown that electronically inferred annotations can be reliable, but their performance varies substantially across inference methods, organisms, and annotation types [11]**.** Several quantitative studies illustrate the magnitude of this concern. For instance, Schnoes and colleagues evaluated 37 well-characterized enzyme families across four widely used databases (UniProtKB/Swiss-Prot, GenBank NR, UniProtKB/TrEMBL, and KEGG) and reported that, while the manually curated Swiss-Prot showed misannotation rates close to zero for most families, the largely electronically annotated databases showed average misannotation rates ranging from 5% to 63% across superfamilies, with more than 80% misannotation in 10 of the 37 families [17]. Similarly, Jones and colleagues estimated that GO annotations propagated via sequence-similarity transfer (ISS-style annotations) had an error rate of

approximately 49%, compared with 13–18% for non-similarity-based annotations [18]. Common error modes include over-prediction, in which a specific molecular activity established for one family member is incorrectly transferred to a homolog with divergent function; loss of specificity, where annotations drift toward overly broad parent terms; and propagation of legacy errors, whereby a single mis-assigned ancestor seeds incorrect annotations across many descendant proteins [11,17]. These error modes are particularly problematic when broad domain- or homology-based assignments are reused as input for downstream computational analyses, because incorrect annotations can compound across propagation steps.

Importantly, these methods often yield broader, less specific annotations than those derived directly from experimental evidence in the primary literature [11,12,14,17,18]. This distinction matters because broad automatic transfer is not equivalent to evidence-based curation from newly published text. Methods based on domains, rules, or phylogenetic propagation are effective for extending known functional signals, but they do not directly address the problem this thesis tackles: identifying, extracting, and justifying GO annotations from the latest biomedical literature. For that reason, there remains a need for approaches that do not simply propagate existing functional signals from homology, domains, or pre-written rules, but instead support literature-grounded functional curation.

1.5 Text Mining, Artificial Neural Networks (ANNs), and Large Language Models (LLMs) in Biomedical Research

Text mining is the computational extraction of structured information from unstructured natural-language text. In biomedical research, it is often used to identify entities such as genes, proteins, diseases, and chemicals in publications, detect relationships among them, and surface

candidate evidence for downstream curation tasks [8]. At its core, text mining requires a way to score how relevant a given document, term, or passage is to a query. A long-standing and still widely used approach is the bag-of-words representation, in which a document is treated as an unordered collection of its constituent terms; word order and grammatical structure are discarded, and only term identity and frequency are retained [19]. Although deliberately simplistic, this representation is computationally efficient and forms the basis of three closely related weighting schemes that underpin most classical retrieval pipelines. Term Frequency (TF) measures how often a query term appears in a document, on the assumption that more frequent terms are more representative of its content. Term Frequency–Inverse Document Frequency (TF-IDF) refines this idea by down-weighting terms that occur in many documents and are therefore less informative, so that distinctive terms contribute more to the relevance score than common ones [19,20]. Best Matching 25 (BM25) extends TF-IDF with two additional adjustments: a saturation function that prevents the score from growing without bound for very high term frequencies, and a length normalization that compensates for documents being longer or shorter than the corpus average [21]. Despite their reliance on this simplified representation, these methods remain strong baselines for biomedical retrieval [8].

Bag-of-words methods are effective at retrieving relevant documents, but they have a clear limitation: they treat words as isolated symbols and do not capture meaning. The phrases "induces apoptosis" and "promotes programmed cell death" describe the same biological event, yet a bag-of-words system sees them as nearly unrelated because they share few words. To bridge this gap between surface vocabulary and underlying meaning, biomedical text mining has increasingly turned to machine learning. In machine learning, a model is not given an explicit set of rules for solving a task. Instead, the model contains many internal numerical parameters that are gradually adjusted by examining many input–output examples until it can produce useful outputs for new inputs it has not

seen before [22]. The practical advantage of this approach is that the model can discover regularities in the data that would be difficult for a human to specify by hand, including meaning-level similarities that bag-of-words representations miss.

One particularly successful family of machine-learning models is the artificial neural network (ANN). An ANN consists of layers of simple computational units, often called artificial neurons, connected by numerical weights. Each neuron receives values from the previous layer, combines them according to its weights, applies a simple non-linear transformation, and passes the result to the next layer. During training, the weights are repeatedly adjusted so that the network's outputs approach the desired outputs on the training examples [22]. When ANNs are applied to text, they typically learn dense numerical vectors, called embeddings, that represent each word, phrase, or sentence as a point in a high-dimensional space. Crucially, these embeddings are arranged so that words and phrases with similar meaning end up close to one another in this space, even when they share no letters. This property allows ANN-based systems to recognize that "apoptosis" and "programmed cell death" refer to related concepts, which bag-of-words methods cannot do directly. The way an ANN processes text, however, depends strongly on its architecture, and different architectures handle sequential information in different ways. Because language is inherently sequential — the meaning of a word often depends on the words that came before it — early ANN-based text-processing systems were built around the recurrent neural network (RNN). An RNN reads a sequence one token at a time, where a token is a small unit of text such as a word, subword, or punctuation, and maintains an internal "hidden state" that summarizes everything it has read so far. Each new token updates this hidden state, so the network's interpretation of the current word can take into account the words that preceded it. In its plain form, however, an RNN tends to "forget" information from tokens that appeared many positions earlier in the sequence. The long short-term

memory network (LSTM) was developed specifically to address this limitation by introducing a system of internal gates that determine which information from the hidden state should be retained, updated, or discarded as the network processes the sequence [23]. For much of the 2010s, LSTMs and related recurrent architectures were the standard ANN tools for text-based tasks, including biomedical natural language processing (NLP). The combination of dense embeddings and recurrent processing produced a measurable step change in performance on practical biomedical extraction tasks. For example, Habibi and colleagues trained a hybrid version of LSTMs (LSTM-CRF) for biomedical named-entity recognition, using word embeddings derived from PubMed and PubMed Central (PMC). The system achieved F1 scores of roughly 78–82% across multiple gene, chemical, and disease corpora, outperforming the hand-crafted, entity-specific tools that had been standard in the field at the time [24].

Recurrent architectures such as LSTM were largely superseded by the transformer, a more recent ANN designed to process sequences of text. Unlike recurrent networks, which read tokens one after another, transformers process an entire sequence in parallel and rely on an attention mechanism to weigh each token's relevance to every other. In simple terms, attention allows the model to "look back" across a sentence and decide which words matter most when interpreting a given word [25]. This architectural change brought two practical benefits: transformers capture long-range dependencies between words more effectively than earlier networks, and they scale far more efficiently to large datasets and model sizes. These properties are the main reason transformers have become the dominant architecture for modern language models [10,11]. An early biomedical application of this architecture is BioBERT, a transformer-based language model pre-trained on PubMed abstracts and PMC full-text articles. BioBERT improved performance on biomedical named-entity recognition by 0.62% in F1 score, on biomedical relation extraction by 2.80% in F1 score, and

on biomedical question answering by 12.24% in mean reciprocal rank when evaluated on standard biomedical benchmarks [26]. BioBERT, therefore, illustrates that transformer-based ANNs, when adapted to the biomedical domain, can outperform both classical text-mining methods and earlier ANN architectures on the same tasks.

Large language models (LLMs) extend this trajectory by scaling transformer-based ANNs to billions of parameters and training them on vast, diverse text corpora. At a high level, they learn by predicting the next token in a sequence. By repeating this task across enormous volumes of text, the model gradually builds statistical knowledge of grammar, terminology, common reasoning patterns, and associations between concepts. This stage is called pretraining: the model is first exposed to broad text data to acquire general language competence before being adapted to more specific tasks [27,28].

On the one hand, their relevance to GO curation is straightforward. Functional evidence is typically reported in prose rather than in a structured ontology format. An LLM can, in principle, read a passage describing an experiment, infer the biological activity or process reported, and determine whether that signal supports a candidate GO term. This makes LLMs appealing for tasks that fall between information retrieval and formal annotation. Recent biomedical studies have already used LLMs in medical and genomics-oriented benchmarks, including patient-oriented clinical question answering and cancer variant interpretation, highlighting both the promise and the current limitations of these models in biomedical settings [29,30].

On the other hand, unoptimized LLMs are insufficient for curation-grade use. A particular concern is hallucination, defined in the natural-language-generation literature as the production of content that is neither faithful to nor supported by the input or any reliable source: content that may

be fluent and plausible-sounding yet factually incorrect or unverifiable [31]. LLMs may hallucinate unsupported statements, overgeneralize from weak evidence, or generate answers that sound plausible but are not traceable to the source text. This problem is particularly serious in biomedical applications, where incorrect but confident outputs can mislead downstream interpretation [32,33]. For GO term annotation, reliability matters as much as coverage, and annotation systems must indicate the source of a claim and adhere to a clear logical framework.

1.6 Improving LLM Reliability for GO Curation

1.6.1 Retrieval

Curating GO annotations is challenging because the GO search space is large, with nearly 40,000 GO terms. Asking an LLM to map raw text directly against thousands of possible GO terms increases both computational cost and the likelihood of hallucinated or weakly grounded outputs [2,3]. Classical text-mining methods such as TF-IDF and BM25 provide an efficient first-pass mechanism for reducing this search space by prioritizing GO terms that are lexically and statistically plausible given the input text [19,20,21].

In a hybrid system, retrieval constrains generation. Rather than asking the model to reason over the entire ontology, the system first provides a narrowed set of candidates. This makes the subsequent reasoning step more efficient, more interpretable, and easier to validate. The same general logic is supported by broader LLM research showing that external retrieval improves performance when knowledge is sparse, long-tail, or insufficiently encoded in model parameters [32]. For instance, a 2025 systematic review and meta-analysis found that retrieval-augmented generation

(RAG) improved the overall performance of biomedical LLMs relative to baseline non-RAG systems. The review also summarizes concrete cases in guideline interpretation, domain-specific reasoning, and clinical information extraction [33]. More specific studies reported that retrieval augmentation improved medical reasoning in a bioinformatics framework called Self-BioRAG and enhanced the extraction and summarization of information from electronic health records [34,35].

1.6.2 Open-Weight, Decoder-Only Models

For systems used in research, development, and domain adaptation, open-weight models offer practical advantages over proprietary systems. They can be run locally, which is important when users wish to avoid sending unpublished documents or internal datasets to external servers. They also support repeated offline experimentation, direct fine-tuning, and closer inspection of the architecture and technical documentation. These properties make them especially attractive in research settings where reproducibility, privacy, and model customizability are important [36,37].

The transformer architecture introduced by Vaswani and colleagues comprises two components. The encoder maps an input sequence to an internal numerical representation, and the decoder autoregressively generates an output sequence, producing one token at a time conditioned on the encoder representation and previously generated tokens [25]. Subsequent architectures retain only one of these components. Encoder-only models, such as BioBERT, discussed in Section 1.5, are optimized for tasks that require interpreting existing text, including classification, named entity recognition, and relation extraction [26]. Decoder-only models retain only the decoder and generate text autoregressively from an input prompt, without a separate encoding stage. This design underlies most contemporary LLMs [27].

Two prominent open-weight, decoder-only model families are Llama-3 and Qwen3. The Llama-3 family, released by Meta AI, comprises a series of open-weight, decoder-only transformer models pre-trained on approximately 15 trillion tokens of multilingual text [36]. The public release includes detailed documentation of the pre-training data and the post-training procedure used to align the model with user instructions. The Qwen3 family, released by Alibaba, is a more recent series of open-weight, decoder-only transformer models with greater emphasis on multilingual coverage and tasks requiring explicit step-by-step reasoning [37]. A distinguishing feature of Qwen3 is its dual-mode operation: in "thinking" mode, the model produces an intermediate chain of reasoning before its final answer, whereas in standard mode it generates the final answer directly. Compact 8-billion-parameter (8B) variants of both families (Llama-3.1-8B and Qwen3-8B) have been widely adopted because they can be fine-tuned and deployed on a single research-grade GPU, making them practical starting points for domain adaptation studies — that is, studies in which a general-purpose LLM is further trained on specialized text corpora (for example, biomedical literature, legal documents, or financial reports) so that it performs more reliably on tasks within the chosen domain [36,37].

Although direct biomedical literature on the exact Qwen3-8B checkpoint remains limited, given its recent release, the broader Llama and Qwen families have already been evaluated in biomedical studies. For example, MeDiSumQA evaluated Llama-3.1 for patient-oriented medical question answering, and Lin et al. benchmarked Llama-3.1 and Qwen 2.5 for cancer genetic variant classification [29,30]. These evaluations underscore the practical relevance of the Llama and Qwen families as starting points for biomedical domain adaptation.

1.6.3 Prompt Engineering and Fine-Tuning

Prompt engineering involves designing clear, structured inputs that help an LLM perform tasks more consistently. In biomedical applications, prompt design matters because tasks often require the model to follow strict instructions, distinguish relevant from irrelevant evidence, and produce results in a controlled format. Well-structured prompts can significantly enhance LLM performance on specialized biomedical tasks [38].

Fine-tuning goes a step further. Rather than changing only the input wording, supervised fine-tuning updates the model itself with task-specific data. This enables a general-purpose LLM to adapt to the vocabulary, reasoning style, and output structure required for a domain-specific task. In biomedicine, this is especially valuable because the model must often interpret specialized terminology and produce evidence-based outputs rather than generic summaries. BioInstruct is a strong example: Tran et al. created a large biomedical instruction dataset and demonstrated that instruction tuning improved performance across multiple biomedical NLP tasks [39].

Despite its effectiveness, full-parameter fine-tuning of modern LLMs is often prohibitively expensive because these models contain billions of numerical parameters (“weights”) that encode learned patterns from data. Fine-tuning such models in their original full-precision form typically requires substantial computational resources, limiting accessibility and slowing iterative experimentation. Parameter-efficient fine-tuning (PEFT) methods address this limitation by updating only a small subset of trainable parameters while keeping most pre-trained weights frozen. One widely adopted PEFT approach is Low-Rank Adaptation (LoRA), which injects trainable low-rank matrices into selected layers of the model, enabling efficient task adaptation with minimal additional parameters and a reduced memory footprint [40].

More recently, Quantized Low-Rank Adaptation (QLoRA) has emerged as a practical extension of LoRA, combining parameter-efficient adaptation with aggressive weight quantization. In QLoRA, the original model weights are frozen and stored in a 4-bit quantized format, so each weight is represented by a small, discrete set of values (e.g., 16 levels such as 0.125 or –0.875) rather than high-precision floating-point numbers (e.g., 0.137492 or –0.842761). This substantially reduces memory usage and computational cost while preserving the original model's expressive capacity. Rather than modifying the quantized base model directly, QLoRA introduces small, trainable, low-rank adapter matrices that learn task-specific adjustments during fine-tuning. These adapters effectively compensate for information loss from quantization, enabling performance comparable to full-precision fine-tuning. As a result, QLoRA enables fine-tuning LLMs on modest hardware without sacrificing accuracy or stability [41].

1.6.4 Human Feedback

Another strategy for improving LLM output is to use human feedback. In this setting, people review model outputs and indicate which responses are more useful, accurate, or appropriate. These judgments can then be used to further adapt the model to better match the intended task. The best-known version of this idea is reinforcement learning from human feedback (RLHF), which has become widely recognized as a method for improving instruction-following and overall helpfulness in LLMs [28].

In a classical RLHF pipeline, the model first generates multiple candidate responses to the same prompt. Human annotators then rank or score these responses. The collected preferences are used to train a separate reward model that learns to predict which outputs humans prefer. The

language model is then further optimized via reinforcement learning to produce responses with higher predicted reward. In practice, this can make the model better at following user intent rather than simply generating statistically likely text [28].

A related, simpler preference-learning method is direct preference optimization (DPO). Like RLHF, DPO uses human preference data but avoids the need for a separately trained reward model and a full reinforcement learning loop. Instead, the model is optimized directly on pairs of preferred and non-preferred responses. In plain terms, DPO teaches the model to assign higher probability to responses humans preferred and lower probability to those they rejected [42].

Human-feedback-based alignment is promising, but it must be used carefully in scientific workflows, where helpfulness and preference satisfaction are not substitutes for correctness. A model can become better at producing responses that users prefer without necessarily becoming more factually reliable. In addition, improvements in instruction following may come at the cost of other capabilities, a trade-off that has been described as an alignment tax [43,44].

1.7 Thesis Rationale and Objectives

A central challenge in modern functional genomics is reliably extracting GO annotations from the rapidly expanding biomedical literature. While GO provides a structured, computable framework for representing gene product function, converting textual evidence into curated annotations remains slow, labour-intensive, and difficult to scale. Existing automated approaches are effective at propagating known functional signals from domains, sequence similarity, or evolutionary relationships, but they are less well suited to extracting and justifying GO annotations directly from

newly published text. At the same time, LLMs offer new opportunities for literature-grounded annotation, but their outputs must be carefully constrained and evaluated to ensure reliability, interpretability, and biological relevance. These challenges motivated the development of GOFinder-AI, a retrieval-first annotation framework designed to support accurate, transparent GO annotation from scientific text. GOFinder-AI addresses this problem through a two-stage workflow. First, a pre-query text-mining step reduces the GO search space from thousands of possible GO terms to a much smaller set of high-priority candidates that are plausible for the input text and practical for downstream model evaluation. Second, these candidate terms are assessed via LLM-based inference to determine their relevance to the text. To improve performance and reduce hallucinated or weakly grounded outputs, multiple LLMs were fine-tuned and systematically compared. The resulting framework not only predicts GO-term relevance but also generates annotation cards containing supporting citations, reasoning, and linked biological entities, thereby improving the transparency and interpretability of the annotation process.

To evaluate the practical utility of this approach, GOFinder-AI was used to annotate poorly characterized human genes. For each target gene, relevant biomedical text was collected and processed through the GOFinder-AI pipeline to generate candidate GO annotations with explicit supporting evidence. The resulting annotations were then assessed for biological plausibility and consistency with the cited text, enabling evaluation of the system not only as a classification framework but also as a practical tool for generating evidence-linked functional hypotheses.

We hypothesize that a retrieval-first, fine-tuned LLM pipeline can accurately extract literature-grounded GO annotations from biomedical text, outperform baseline text-mining and zero-shot LLM approaches, and provide a practical framework for the functional annotation of poorly characterized human genes. This hypothesis was addressed through three objectives:

- Objective 1: Extract GO term candidates and their corresponding biological entities from scientific text (e.g., NCBI-PubMed literature).
- Objective 2: Determine whether each extracted GO term is relevant to the input text while generating annotation cards containing supporting citations, reasoning, and linked entities.
- Objective 3: Annotate poorly characterized human genes using GOFinder-AI.

2. Methods

This section outlines the computational workflow of GOFinder-AI. The pipeline was designed to balance two competing goals: reducing the GO search space to a manageable size for efficient LLM inference while preserving sufficient biological coverage for accurate annotation. As summarized in Figure 5, the workflow comprises five stages. First, the user provides input text. Second, the text is processed to identify traceable biological entities and to shortlist the top 100 most likely GO term candidates. Third, the input text, extracted entities, and candidate GO terms are incorporated into an engineered prompt and passed to a fine-tuned LLM. Fourth, the LLM assesses whether each candidate GO term is relevant to the input text and, when applicable, returns supporting citations, explanatory reasoning, and linked entities. Finally, the user reviews the generated annotations and decides whether to save or discard them. The system records these decisions to support future model optimization and iterative improvement. The following subsections describe each stage of the pipeline in detail.

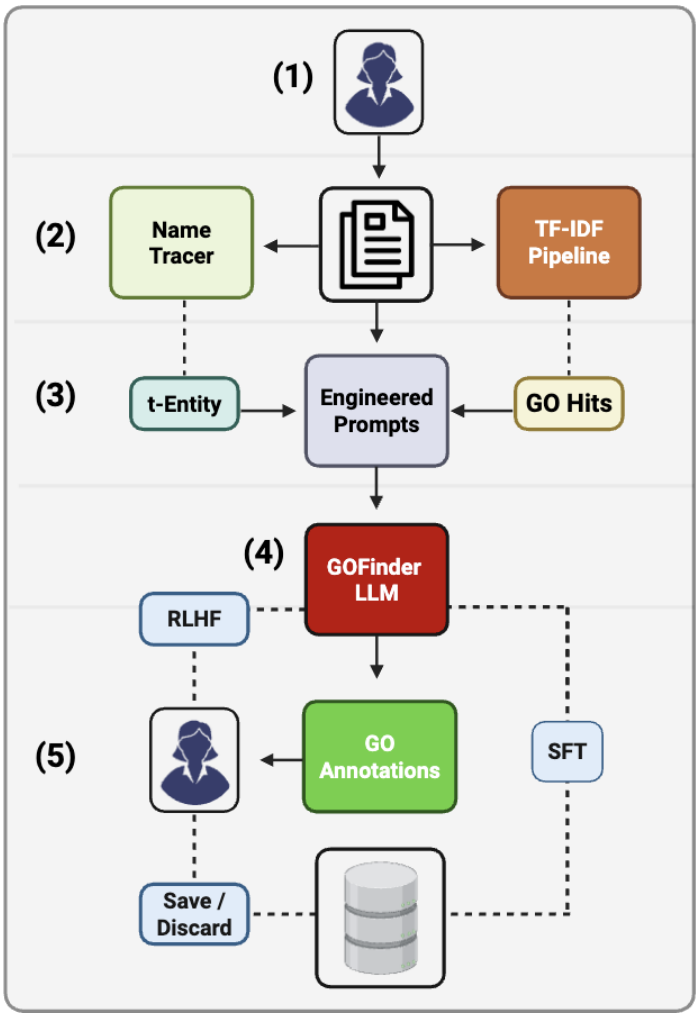

Figure 5. GOFinder-AI Annotation Pipeline. The GOFinder-AI framework uses fine-tuned versions of Llama-3.1-8B and Qwen3-8B to assign GO annotations to biological entities of interest based on literature-derived evidence. The pipeline consists of five main stages. (1) The user provides input text, either directly or as a PDF. (2) The text is processed with Name-Tracer, which identifies all traceable entity names in the document, and with the TF-IDF GO-Prioritization module, which retrieves the top 100 candidate GO terms. (3) The text, traceable entities, and GO candidates are incorporated into an engineered prompt and passed to the LLM. (4) The LLM determines whether each candidate GO term is relevant to the text and, when applicable, returns supporting citations, reasoning, and linked entities. (5) The user then reviews the generated annotations and either saves or discards them. These user decisions are recorded for future reinforcement learning from human feedback (RLHF), while validated annotations support subsequent rounds of supervised fine-tuning (SFT).

2.1 Data Collection and Preprocessing

We filtered a set of 18,890 PMID-Entity-GO annotations from AmiGO to evaluate the performance of the GO-Prioritization and Name-Tracer modules. A subset of 15,091 annotations was used to fine-tune and evaluate the GOFinder-AI pipeline against other LLMs [45,46].

2.2 GO-Prioritization Module

We manually curated a list of 2,252 GO terms that capture the central themes of cellular and protein biology and account for 99.66% of all curated biological process GO annotations in the AmiGO database [45]. We then developed a GO-Prioritization module that preprocesses input text (typed or PDF) to extract keywords and, using a hybrid TF-IDF weighting algorithm, ranks the top GO terms.

2.2.1 Curated GO list

There were 39,906 GO terms, excluding obsolete ones. Of those, 25,699 were BP terms, the category we focused on with our tool. The decision to restrict the present study to BP terms was motivated by two complementary considerations: the level of functional insight that BP terms convey and the comparative difficulty of assigning them through existing automated methods. BP terms (e.g., "intrinsic apoptotic signaling pathway" or "regulation of cell cycle G1/S phase transition") describe the higher-order pathways and programs to which a gene product's activity contributes, and therefore situate molecular activity within the broader biological themes that motivate most downstream interpretation in functional genomics. In this respect, BP annotation conveys a level of functional insight that neither MF terms, which describe biochemical activity at the molecular level, nor CC

terms, which describe subcellular localization, can supply on their own [2,3]. This same property, however, makes BP terms the most difficult of the three GO domains to assign computationally. Škunca and colleagues evaluated electronically inferred GO annotations across all three domains and reported that BP terms exhibit the lowest coverage, MF terms the highest, and CC terms an intermediate level, demonstrating a pattern consistent with BP terms being the hardest of the three to assign and MF terms the easiest [11]. The same asymmetry has been observed in successive Critical Assessment of Functional Annotation (CAFA) challenges, where top-performing predictors consistently achieve lower precision and recall on the BP ontology than on MF or CC [47]. Two structural factors underlie this gap. CC terms recur in relatively consistent prose contexts across the literature: references to entities such as the nucleus, the mitochondrial inner membrane, or the spliceosomal complex tend to appear in stereotyped phrasing, a pattern that makes them comparatively tractable for rule-based and IEA pipelines [6,12]. MF terms, although more granular, remain partially tractable for electronic annotation through curated mappings between Enzyme Commission (EC) numbers and the GO term, in which EC-annotated enzymes in UniProtKB are automatically associated with the corresponding GO MF term and the resulting annotation is supported by the IEA evidence code [6,12]. BP terms, by contrast, describe processes that are typically distributed across multiple sentences of a publication, expressed in heterogeneous prose, and only weakly correlated with sequence- or domain-level features. These characteristics limit the reach of homology-, signature-, and rule-based IEA approaches [11]. Restricting the present study to BP terms, therefore, aligns the scope of the tool with the GO domain, where a literature-grounded, LLM-assisted curation system is most likely to add value over existing electronic annotation pipelines.

Prioritizing GO terms naively from this 25,699-term list using a keyword-based matching script, such as TF-IDF, or even an LLM, is not only computationally inefficient but also highly

unreliable. Therefore, we followed a multi-layered protocol to curate this list. Initially, we used the BP ontology gene sets available in the Molecular Signatures Database (MSigDB) to obtain 9495 GO terms [48]. We further reduced the list by merging redundant terms. Per a reduction decision, we identified the major theme at hand (e.g., apoptosis) and all terms corresponding to that theme. Then, we kept a representative set of GO terms that capture the nuance underlying this theme (e.g., “apoptotic process”, “[Tissue Type] apoptotic process”, “cellular component disassembly involved in execution phase of apoptosis”) and merged redundant terms (e.g., grouping “negative regulation of apoptotic process” and “positive regulation of apoptotic process” child terms under the “apoptotic process” parent term). The merged terms are added to the “Spread Annotation” column corresponding to the primary GO term we kept. These are analyzed by the LLM when the primary GO term is ranked highly by the GO term prioritization module. We also added a curated selection of MF terms that represent the main themes of protein biology (e.g., prominent protein families, such as G protein activity) and that are not conceptually covered by the BP terms. This brought the curated list down to 2,252 GO terms (1982 BP; 270 MF), which, together with their “Spread Annotation” terms, account for 99.66% of all curated GO annotations in the AmiGO database (excluding irrelevant MF and CC annotations), indicating that the list not only covers the major biological themes in cellular biology but is also well represented in the current literature-sourced annotations [45]. For each of the 2,252 GO terms, we manually curated keywords that correspond semantically to the GO term and are well represented across its corresponding annotations (e.g., the keywords curated for the “apoptotic process” GO term included “apoptosis”, “caspase”, “cell-death”, etc.). We also assigned one “priority” keyword to each GO term from its curated keyword pool, such that if this keyword appears in the text, the GO term in question jumps to the top of the candidate list of GO terms by boosting its text similarity score. The priority keyword is defined as the keyword most commonly

associated with a GO term (e.g., “GTPase” is the priority keyword for the GO term GTPase activity (GO:0003924)).

2.2.2 TF-IDF Weighting Algorithm

Before using the TF-IDF script to rank the top 100 GO term candidates, both the GO term keyword sets and the input text undergo a standardized preprocessing pipeline to ensure comparability. The preprocessing module performs several normalization steps: **(i)** conversion of chemical notation to natural-language equivalents (e.g., "Mg^{2+}" to "magnesium ion", "Trp" to "tryptophan amino"); **(ii)** removal of punctuation, statistical notation (e.g., p-values, sample sizes), and non-alphabetic tokens; **(iii)** Greek-letter normalization (converting Unicode symbols to their English equivalents); **(iv)** Porter stemming to reduce morphological variants to common roots (e.g., "differentiation" → "differenti", "phosphorylation" → "phosphoryl"); and **(v)** removal of stop-words using a domain-specific list curated for GO terminology (e.g., "process", "activity", "regulation"). Hyphenated compound terms are decomposed to capture both the compound and its constituent morphemes (e.g., "alpha-beta" yields ["alpha-beta", "alpha", "beta"]).

A critical component of the preprocessing pipeline is lexical expansion, which addresses the challenge of detecting GO-relevant keywords embedded within larger compound terms or biologically relevant morphemes (e.g., "-ase" → “enzyme”). The system employs a curated list of 700 lexicon mappings organized into four matching categories: **(i)** exact matches, where a stemmed token that matches precisely expands into frequently recurring synonyms (e.g., "chemokin" → "signal"); **(ii)** anywhere substring matches, where a morpheme appearing anywhere within a token triggers expansion (e.g., detecting "gtpase" within "RhoGTPase" yields the GO-relevant keyword); **(iii)** word-initial prefix matches, where morphemes at the beginning of a token trigger expansion (e.g.,

"hem-" in "hemoglobin" → "hemo", "blood"); and **(iv)** word-final suffix matches, where terminal morphemes indicate functional categories (e.g., "-ase" in "kinase" or "phosphatase" → "enzym"; "-ose" in "glucose" or "fructose" → "sugar"; "-ol" in "phytol" → "alcohol"). This multi-tiered expansion strategy enables detection of both exact keyword matches (e.g., "GTPase" appearing as an isolated term) and partial matches embedded within compound biological nomenclature (e.g., extracting "GTPase" from "RhoGTPase"). It also reflects common biomedical term construction patterns (e.g., "phospho-" → "phosphoryl"; "glyco-" → "carbohydr", "sugar").

After preprocessing, the keywords from both the GO term (GO) and the text (ABS) are stored as vectors and then ranked by comparing their cosine distance (as shown in Formula 1). Each keyword in a GO term's vector is weighted. TF·IDF (Formula 3) is a standard weighting scheme that multiplies the term frequency (TF) by the Inverse Document Frequency (IDF) (Formula 2). The IDF is calculated in advance and is lower for keywords that are more common in the GO list corpus (e.g., regulation, protein). Therefore, TF·IDF will rank a GO term highly for a given text if its keywords overlap with those extracted from the input text (high TF) and if those keywords are rarer across the GO list corpus (high IDF). BM25 (Formula 4) is a more sophisticated version of TF·IDF that also accounts for the number of keywords in a GO term relative to the average across GO terms. This formulation provides a more robust ranking when the input text varies substantially in length or when specific keywords appear with unusually high frequency.

$$\text{Cosine}(ABS, GO) = \frac{\text{weightedvector(ABS)} \cdot \text{weightedvector(GO)}}{|\text{weightedvector(ABS)}| \times |\text{weightedvector(GO)}|}$$

Equation 1. **Cosine Similarity, between Two Vectors, ABS and GO.** Cosine similarity quantifies the normalized overlap between two weighted keyword vectors, enabling comparison of the abstract (ABS) and GO term representations based on shared keyword structure rather than on absolute frequency.

$$\text{idf}_j = \log_2\left(\frac{\text{number of GO Terms}}{\text{GO Terms with keyword } j}\right)$$

Equation 2. Inverse Document Frequency (IDF) Weighting Equation. Idf(j) calculates the relevance weight of keyword j by computing the logarithmic ratio of the total number of GO terms in the GO list corpus to the number of GO terms containing keyword j. This logarithmic equation assigns a higher weight to keyword j if it appears in fewer GO terms (i.e., high specificity) and a lower weight to ubiquitous terms such as "cell" or "protein," which offer limited discriminative value.

$$\text{tf-idf}_j = \text{tf}_j \times \text{idf}_j \quad \text{where } \text{tf}_j = \text{number of times keyword } j \text{ appears}$$

Equation 3. **Term Frequency-Inverse Document Frequency (TF·IDF) Weighting Equation.** Tf-idf(j) calculates the relevance weight of keyword j by combining tf(j), which represents the raw frequency of keyword j in a given publication, with idf(j), which quantifies keyword j's discriminative power across the GO list corpus.

$$\text{BM25}_j = \frac{\text{tf}_j \times \text{idf}_j \times (k+1)}{\text{tf}_j + k\left(1 - b + b\left(\frac{\text{number of GO Keywords}}{\text{average number of GO Keywords}}\right)\right)} \quad \text{where } k = 1.6,\ b = 0.75$$

Equation 4. **Best-Matching 25 (BM25) Weighting Equation.** BM25(j) calculates the relevance weight of keyword j by combining its raw frequency in the publication, tf(j), with its inverse document frequency, idf(j), while adjusting for document length relative to the average keyword count across GO term entries. The parameter k (default 1.6) controls term frequency saturation, and b (default 0.75) controls the degree of length normalization.

2.3 Name-Tracer Module

A common challenge is the inconsistent use of naming conventions and aliases for genes and proteins in the scientific literature. For example, the gene “AFP” may be referred to in research articles as “alpha-fetoprotein”, “α-fetoprotein”, or by synonymous names such as “alpha-1-Fetoprotein" or “AFPD”. This variability complicates text analysis and GO information extraction. To address this issue, we developed Name-Tracer, a tool that standardizes these alternative

nomenclatures. Name-Tracer is a string-matching script that traces biological entities (genes, proteins, isoforms) in any given text using an extensive list of human gene-level aliases (i.e., synonyms) as its reference. We curated the alias list from publicly accessible annotation resources, including Ensembl BioMart and UniProt [15,49].

Name-Tracer takes two inputs: a user-provided entity name (if applicable) and the corresponding text. The module uses a tiered matching strategy. First, it performs exact matching against the user-provided entity, accounting for standard naming conventions and Greek-letter variants (e.g., Δ, alpha, beta). If no exact match is found, it uses gene-level aliases for exact string matching. Finally, when exact matches fail, approximate string matching is performed using Python's SequenceMatcher with a ≥80% threshold to capture minor lexical variations while avoiding false positives. The traceable entities can then be injected as candidates into the downstream LLM prompt. By anchoring prompts to matched entity names, this preprocessing step reduces ambiguity and constrains the LLM's generation space, thereby mitigating hallucinated GO annotations.

2.4 LLM Prompt Engineering

We optimized the prompt for our task using a general framework commonly applied to biological applications, such as gene set summarization [50]. The prompt was divided into three main components: task definition, strategies for accomplishing the task, and output format. It required the model to confirm familiarity with GO terms and to map GO annotations using explicit evidence from the provided text. The prompt mandated a structured list of GO terms with associated citations and reasoning. Despite its structured approach, the initial prompt exhibited several limitations that hindered its effectiveness. First, the prompt required manually changing the entity name for each

new query, making the process inefficient and error-prone. Second, the output contained an unreasonable number of GO terms, likely due to hallucinations in the model's predictions. Additionally, the generated results often lacked GO term titles, supporting references, and proper reasoning, deviating from the intended output format.

To address these limitations, we introduced a series of modifications to the prompt. First, the model was explicitly assigned the role of a Biomedical Curator, a form of role-based prompting that has been shown to improve the LLM's adherence to the task, domain alignment, and output reliability across diverse tasks [51,52]. The revised prompt enforces a consistent JSON-based structure for both the input and the output. For the input, the top 100 GO candidates ranked by the GO-Prioritization module, along with their metadata (name, definition, and synonyms), are injected into the prompt via variable replacement, and the traceable entity names identified by the Name-Tracer module are included. The prompt specifies a fixed output schema and includes manually curated examples that demonstrate valid annotation formats. Example-based prompting has been shown to significantly improve task performance by anchoring the model to expected reasoning patterns [27]. Additionally, the model was instructed to reason explicitly, step by step, using chain-of-thought prompting, a technique that improves factual correctness and interpretability in complex reasoning tasks by encouraging intermediate reasoning steps rather than direct answer generation [53].

2.5 LLM Selection and Fine-Tuning

Two open-weight models were used for fine-tuning and comparative evaluation in this study: Llama-3.1-8B and Qwen3-8B (Section 1.6.2). Both are decoder-only models, which suits the requirements of the GOFinder-AI pipeline. For each candidate GO term, the pipeline must first

determine whether the term is relevant to the input text and, if so, produce supporting citations, a short reasoning statement, and the linked biological entities. An encoder-only model, such as BioBERT, could perform only the relevance decision, whereas a decoder-only model can carry out both steps in a single pass. Llama-3.1-8B and Qwen3-8B were selected within this class because they share a similar size (8 billion parameters) and can each be fine-tuned and run on a single research-grade GPU. Their similar scale allows them to be compared directly under the same fine-tuning, evaluation, and deployment conditions, while the two models themselves represent complementary design choices: Llama-3.1-8B is an established instruction-following model, whereas Qwen3-8B is a more recent reasoning-oriented model. They were chosen as practical, well-documented starting points for biomedical adaptation rather than as models known in advance to be optimal for GO curation; the cross-comparison we reported in Section 3 is intended to test which of the two performs better on this specific task.

Using Llama Factory's 4-bit QLoRA method, we fine-tuned the Llama-3.1-8B and Qwen3-8B models on an AmiGO-sourced dataset of 22,964 gene-level GO annotations [36,37,45,54]. The model was trained to identify relevant GO term candidates from any given text and to generate annotation cards that include supporting citations, reasoning, and linked entities (e.g., genes, proteins, or isoforms associated with the GO term). More specifically, a set of 59,754 gene-level annotations was sourced from the AmiGO database, and the entities were traced with confidence using the Name-Tracer module. We then filtered this set to 15,091 annotations with curated GO terms that were detected in the top 100 list of GO term candidates generated by the GO-Prioritization module, indicating that there are GO keyword signals in the PubMed text corresponding to these annotations. For each of the 15,091 annotations, we generated a false annotation with a GO term that is the most hierarchically distant from the annotated GO term (i.e., the term with the most distant

lowest common ancestor) and that is semantically unrelated to the PubMed-sourced text, with similarity scores at or below 15% reported by the TF-IDF script used in the GO-Prioritization module. This last step generated 15,091 false annotations, bringing the total to 30,182. These annotations correspond to 9,705 PMIDs, 75% (7,279 PMIDs, 22,964 annotations) of which were used to fine-tune the LLMs and 25% (2,426 PMIDs, 7218 annotations) of which were used to run a final test to cross-compare the LLMs with examples unseen during the fine-tuning. We ran a 4-fold grouped cross-validation on the 22,964-annotation fine-tuning set, prior to running a final cross-comparison test against other LLMs. Cross-validation was chosen to provide a more stable estimate of generalization than a single train/validation split and to assess whether performance remained consistent across different partitions of the training pool [55]. To prevent document-level information leakage, we ran the cross-validation at the PMID level. As mentioned above, the 9,705 unique PMIDs represented in the 30,182-sample master dataset were first split into a 75% training pool (7,279 PMIDs; 22,964 annotations) and a 25% held-out final test set (2,426 PMIDs; 7,218 annotations). Only the 75% training pool (7,279 PMIDs) was used during cross-validation. These PMIDs were then partitioned into four mutually exclusive folds using a greedy balancing procedure that distributed PMIDs across folds while keeping total row counts nearly equal. For each cross-validation iteration, one fold served as the validation set and the remaining three folds served as the training set, yielding approximately 17,222 training annotations and 5,741 validation annotations per run. This grouped design ensured that all annotations derived from a given publication remained in the same fold, thereby preventing related examples from appearing in both training and validation sets.

The training datasets were used to fine-tune and evaluate the LLMs on a narrow classification task: determining whether a candidate GO term is IS_RELEVANT or NOT_RELEVANT to a given input text. In supervised fine-tuning, the model is trained to increase the probability of the correct

output sequence given the input prompt. For causal language models, this objective is typically measured using token-level cross-entropy loss (see Formula 5 below), which penalizes the model when it assigns a low probability to the correct next token. In the present task, loss was computed only over the target output tokens (e.g., IS_RELEVANT: TRUE), while the prompt tokens were masked and therefore did not contribute to the optimization objective.

$$L = -\sum_{t \in y} \log P(y_t \mid y_{<t}, x)$$

Equation 5. **Token-Level Cross-Entropy Loss Formula.** In this expression, L denotes the total training loss, t indexes each target output token, and Y represents the set of output tokens over which the loss is computed. yt is the correct token at position t, y<t denotes all previously generated output tokens, and x represents the input prompt, which in this study includes the biological entity, GO-term metadata, and input text. The term P(yt |y<t ,x) is the probability the model assigns to the correct next token given the prompt and prior output tokens. The negative log penalizes low probabilities assigned to correct tokens, so a lower loss indicates better prediction of the target output sequence. In the present task, the loss was computed only over the classification output tokens, such as IS_RELEVANT: TRUE, while the prompt tokens were masked and therefore did not contribute to training.

During training, the training loss indicates how well the model fits the examples it is directly learning from, whereas the evaluation loss measures performance on held-out validation examples that are not used to update parameters. Monitoring both curves is important because a falling training loss alone does not guarantee good generalization. If the evaluation loss stops improving or begins to rise while the training loss continues to decrease, this may indicate overfitting, meaning the model is becoming increasingly specialized to the training examples and losing some ability to generalize to unseen data. To reduce this risk, we added early stopping to our grouped 4-fold cross-validation

fine-tuning procedure: training terminates once validation performance no longer improves for a predefined number of evaluation rounds [56].

Several additional safeguards were used to improve training stability and reduce overfitting. First, training was limited to two epochs, meaning the model passed through the training dataset only twice. This conservative choice was intended to allow task adaptation while reducing the risk of overfitting to the training data. Second, gradient clipping was applied with a maximum gradient norm of 1.0. Gradients determine how strongly model parameters are updated at each training step; clipping limits unusually large updates and helps prevent unstable optimization, especially when fine-tuning large models [57]. Third, a cosine learning-rate schedule with warmup was used. The learning rate is the size of each parameter update during optimization. A high learning rate can make training unstable, whereas a very small one may substantially slow learning. In the present configuration, training began with a short warmup phase, during which the learning rate gradually increased from a small value to its peak. This was followed by cosine decay, in which the learning rate gradually decreased over the remainder of training. The purpose of this schedule was to stabilize the early phase of optimization and then allow increasingly smaller, more conservative updates as training progressed [58].

Both Llama-3.1-8B and Qwen3-8B were fine-tuned with the same QLoRA configuration to enable a fair comparison. During training, the model's parameters are updated repeatedly so its predictions converge toward the correct outputs. This process is controlled by an optimizer, which determines how the model should adjust its internal parameters after each training step. In this study, we used AdamW, a widely used optimizer in deep learning that adapts the size of parameter updates based on the recent training history and applies weight decay, a regularization strategy that helps prevent the model from overfitting to the training data [59]. The initial learning rate was set to 2 ×

10^{-4}, consistent with prior QLoRA-style fine-tuning settings for LLMs, where relatively small but still effective update steps are preferred to preserve pretrained knowledge while allowing task adaptation [41]. Evaluation was performed every 100 steps. Early stopping was triggered after three consecutive evaluations without improvement in validation loss, and the checkpoint with the lowest validation loss was retained. In summary, these settings were chosen to balance efficient adaptation against overfitting: warmup stabilized the start of training, cosine decay made later updates progressively smaller, gradient clipping reduced unstable jumps in parameter values, and early stopping prevented unnecessary continuation once validation performance had plateaued.

2.6 Two-step Integration of Fine-Tuned LLMs

The AmiGO-sourced annotations lack explicit supporting citations or human-readable reasoning for each one. As a result, this dataset was used to fine-tune and evaluate the LLMs on a narrower classification task: determining whether a candidate GO term is IS_RELEVANT or NOT_RELEVANT to a given input text. Once this relevance decision is made, the base model's broader generative capabilities are used to produce supporting citations, explanatory reasoning, and linked entities associated with the predicted annotation.

In practice, for each candidate GO term, the fine-tuned LLM receives the input text and the term’s metadata, then predicts whether the term is relevant to the text. The resulting relevance verdict, along with the original input, is passed to the corresponding base model, which generates supporting citation spans, explanatory justifications, and linked biological entities. This two-stage design was adopted because the fine-tuned models outperformed their base-model counterparts on the specific task for which they were trained, namely GO-term relevance classification, but performed worse on untuned generative subtasks. This behaviour is consistent with reports of catastrophic

forgetting, a phenomenon in which a model adapted to a new task loses part of the general knowledge or broader capabilities acquired during pretraining. In the context of LLMs, this can occur when task-specific fine-tuning improves performance on the target task but degrades performance on untuned tasks, such as open-ended reasoning, knowledge recall, or instruction following, outside the fine-tuning distribution [60]. Recent work has shown that this trade-off can persist even under PEFT methods such as QLoRA, indicating that restricting the number of updated parameters does not fully prevent loss of prior capabilities [60,61]. This problem extends beyond simple forgetting: domain-specific fine-tuning may also reduce the model's ability to integrate its original general-purpose skills with newly learned specialized knowledge, a concern that is especially relevant in workflows such as GOFinder-AI, where both narrow classification accuracy and broader generative abilities are needed within the same pipeline [62].

This two-step integration allows the system to retain the advantages of task-specific fine-tuning while avoiding degradation on downstream subtasks that were not directly represented in the fine-tuning data. In the current version of the tool, the fine-tuned model is used only for relevance classification, whereas the base model is retained for citation extraction, reasoning generation, and linked-entity reporting. This should be regarded as an interim design choice rather than a final architecture. As user adoption increases, the system is designed to support a reinforcement-learning-from-human-feedback framework, in which user evaluations can be incorporated to iteratively improve model performance.

2.7 LLM Performance Optimization with vLLM

One practical challenge in running LLMs locally is computational efficiency, particularly GPU memory usage and inference speed. To improve local model serving efficiency, we used vLLM, an optimized inference engine for LLMs [63]. vLLM improves throughput and memory utilization by managing the model's attention cache more efficiently via PagedAttention, which reduces memory waste during text generation and enables more requests or longer contexts to be handled under the same hardware constraints [63]. In practice, this tool enabled us to run our LLMs locally efficiently and to support the two-step integration of fine-tuned models into our tool.

2.8 Web Interface

We developed a React-based web interface that enables users to interact with the pipeline by providing input text and saving or discarding annotations, thereby collecting human feedback for reinforcement learning and expanding our GO annotation database of poorly annotated genes [64]. The tool supports text in both typed and PDF formats. PDFs are converted to text using the GitHub Unstructured library, which supports heterogeneous publisher layouts and document encodings [65]. After converting the uploaded PDF to text, we apply an additional text-cleaning layer to reduce the amount of text sent to the LLM for processing. This cleaning layer performs line-level filtering to remove journal citations, acknowledgements, and publisher boilerplate, which commonly fragment into low-quality, irrelevant sentence tokens. The interface includes a built-in PDF viewer and highlighting feature, allowing users to trace the supporting citations for each annotation in typed or PDF text and to eliminate any annotations without explicit evidence.

The interface supports multiple modalities, including the primary modality described in Figure 5, which generates annotations for any given text, and a GO Automate modality that connects our

tool to the PubMed interface and generates GO annotations for any entity of interest using relevant publications. This modality enabled us to trace 30 novel annotations for poorly annotated hallmark genes [66]. For GO Automate, we implemented a rule-based retrieval pipeline, PubMed Extract, to identify relevant publications for any given biological entity. First, we query PubMed via the NCBI Entrez E-utilities API for at least one entity mention [66]. To capture all entity mentions, we use the gene-level aliases list curated for the Name-Tracer module. Candidate publications are then scored and ranked using a predefined set of criteria applied to titles and abstracts. These criteria include the frequency of entity mentions and the presence of functional terms. We also consider the journal's impact factor and the publication's recency. This scoring framework prioritizes publications most likely to provide functional evidence relevant to downstream mapping of GO annotations. PubMed Extract uses the Crossref API to filter out retracted publications and track updates or corrections to previously processed PMIDs [67]. Publications previously used by the pipeline that have been flagged as retracted or updated are automatically marked for re-evaluation in subsequent annotation cycles. For this modality, the tool also uses the DOIs of candidate PMIDs to retrieve corresponding full-length PDFs by connecting to several open-access (OA) APIs, including Europe PMC, Unpaywall, and the PMC Open Access Web Service. PMIDs for which OA retrieval fails across all sources are logged for manual handling by the user [68,69,70].

2.9 LLM Direct Preference Optimization (DPO)

To further improve annotation quality after supervised fine-tuning, we use the TRL (Transformer Reinforcement Learning) library to implement a framework that iteratively refines our LLM’s behaviour using validated user interactions [42,71]. In this framework, feedback is collected

during routine tool use through two primary user actions. First, when a generated GO annotation card is explicitly saved, the annotation is treated as a positive signal, indicating that the predicted GO-term relevance, supporting evidence, and linked entities are acceptable for downstream use. Second, when an annotation is explicitly ignored or rejected, the user is prompted to provide a reason for the inaccuracy, and the example is treated as a negative signal. In this way, user interactions are converted into preference information that reflects real annotation utility rather than relying solely on synthetic proxy labels.

Because raw user feedback may be noisy, incomplete, or inconsistent, examples are not incorporated into model updates immediately. Instead, after user consent, batches of approximately 1,000 annotation interactions are transferred for laboratory review and manually confirmed before being admitted to the preference-training pool. This manual review step ensures that only high-quality feedback is used for post-training and reduces the risk that incorrect user actions, interface misuse, or ambiguous annotation cases could adversely skew model performance. Because DPO operates on preferred-versus-rejected response pairs, each curated user feedback record is converted into a preference pair comprising a chosen and a rejected response for the same annotation task. An annotation that is saved and confirmed as accurate is treated as the preferred example, whereas an annotation that is explicitly ignored and confirmed as inaccurate is treated as the rejected example. The model is then updated to increase the likelihood of preferred outputs relative to rejected ones in otherwise matched task contexts.

However, preference updates are not automatically deployed after each alignment cycle. Instead, after each round, the updated model is evaluated on a held-out test set of 7,218 annotations that were not seen during supervised fine-tuning and are never used for reinforcement learning. This evaluation set serves as an external checkpoint to verify that preference optimization has not

degraded core annotation performance. Only if performance on this unseen benchmark is maintained or improved is the updated model accepted into the production pipeline. If test-set performance declines, the update is rejected, and the prior stable model is retained. This safeguard is included to avoid the risks of the alignment tax [43,44].

2.10 Pipeline Evaluation

From the AmiGO database, we extracted all PMID-based annotations corresponding to the "2,252" Curated GO list [45]. We then filtered out publications lacking an abstract (our base evaluation text) and any PMIDs that had been retracted. When applicable, we also replaced outdated PMIDs (i.e., those that had been corrected) with their more recent versions. This left us with a dataset comprising 18,890 PMID-ENTITY-GO annotations, with 14,537 unique PMIDs. The decision to manually curate the list of keywords for each GO term in our dataset resulted in 13,658 of the 14,537 total PMIDs (94%) being traceable for GO keyword signals using only their abstracts. A total of 6% of PMIDs (n=879) showed no abstract-level signal. We manually inspected the full-length text of each of those publications. If there were hits in the full-length text (i.e., sentences containing the GO keyword signals), we added these signal-containing sentences to the text analyzed for the PMID.

We evaluated the GO-Prioritization module on a set of 18,890 PMID-Entity-GO annotations, achieving 89% recall for the top 100 GO terms. We also evaluated the Name-Tracer module on the same set, achieving 87% recall for traceable entities. We then cross-compared the fine-tuned Llama and Qwen with their base versions and the proprietary GPT-5-mini on 7,218 gene-level annotations unseen during fine-tuning. Our fine-tuned Qwen model achieved the best performance, at 95%, in

correctly identifying the relevant GO terms from text when paired with the pre-query GO-Prioritization and Name-Tracer modules [36,37,72,73].

3. Results

3.1 GO-Prioritization Module achieves High Accuracy

On 18,890 annotations, TF-IDF achieved the best performance, with a Recall@100 of 0.89, indicating that 89% of curated GO annotations were recovered within the top 100 ranked predictions (Figure 6). By comparison, BM25 (Recall@100 = 0.84), Two-Step Ranking (Recall@100 = 0.81), and the IDF-only metric (Recall@100 = 0.73) performed less well, though all substantially outperformed the Dummy baseline (Recall@100 = 0.08). As expected, recall for all methods approached 1.0 as K approached the full GO term corpus size (2,252 terms).

Zero-score annotations are those for which the curated GO term receives a zero-similarity score from the GO-Prioritization pipeline. IDF-based methods often share the same zero-score annotations. Through manual inspection, we identified several reasons these annotations received zero similarity scores: the module may fail to match a valid signal in the text (e.g., failing to detect "GTPase" in "RhoGTPase"), which we addressed by modifying the module to trace keyword hits anywhere in the text, including as part of other words. Some cases result from missing keywords during list curation (e.g., "leuko" as a keyword for "immune response"). We resolved these by manually expanding the keyword list assigned to the problematic GO term. Other annotations lacked a GO term signal in the PMID-corresponding abstract, so we addressed this by inspecting the full-length publication and adding the missing signal-containing text, if applicable. Lastly, some

annotations lacked explicit citations in the papers supporting the annotated GO terms. We removed these implicit annotations from the evaluation set. We included BM25 with and without the original zero-score set to examine the impact of our manual modifications to the module. The high overlap between BM25 and BM25 (no zeros) indicates that our approaches have resolved most zero-score annotations. Initially, 1,738 zero-score annotations were identified for the 2,252 Curated GO list. Following manual curation, this number dropped to 331 annotations that lack explicit citations in their full-length papers.

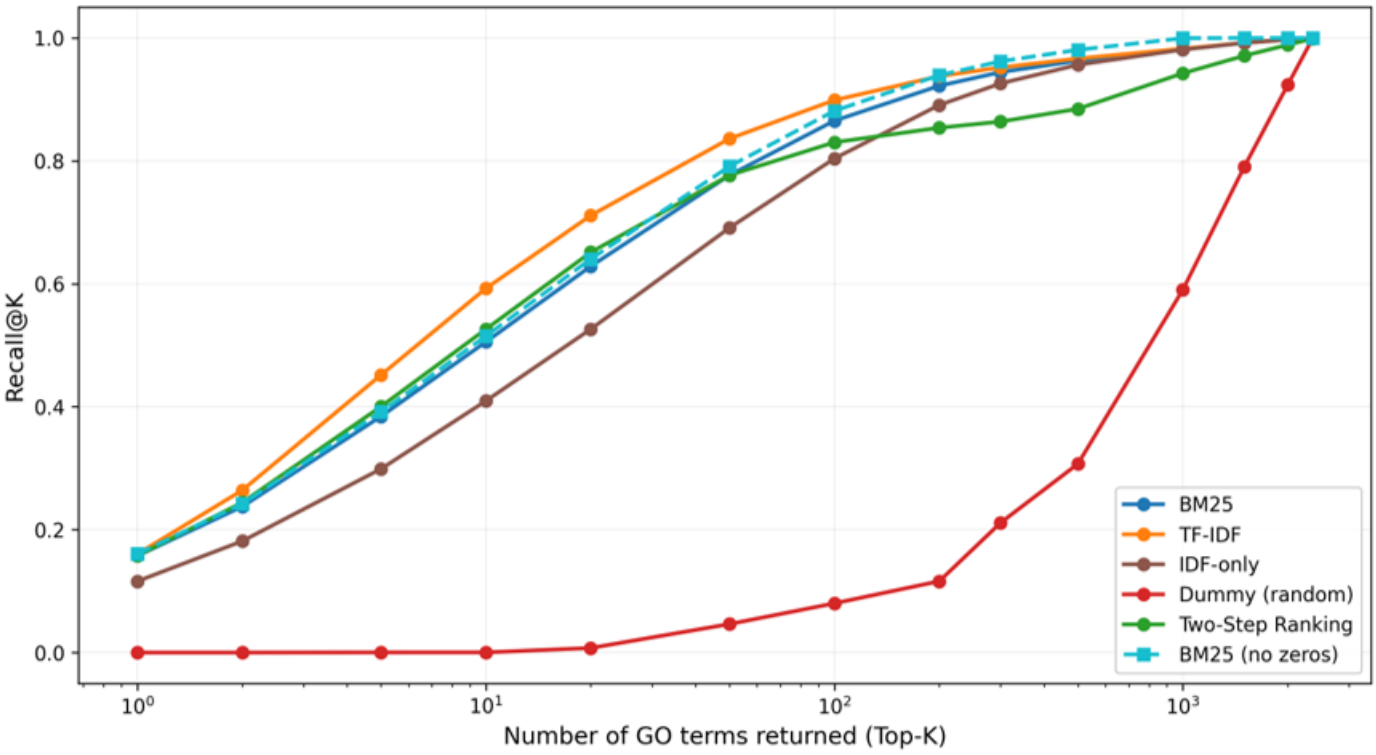

Figure 6. GO-Prioritization Scripts Cross Comparison (Recall@K). Performance of six GO term ranking methods evaluated on 18,890 PMID-ENTITY-GO annotations, with K representing the number of top-ranked GO terms returned (x-axis, log scale). Methods compared include: BM25 (blue), TF-IDF (orange), IDF-only (brown), Dummy random baseline (dark red), Two-Step Ranking (green), and BM25 excluding zero-score annotations (cyan dashed). Zero-score annotations are those in which the curated GO term receives a similarity score of zero from the GO-Prioritization pipeline. The Two-Step Ranking method uses a hybrid strategy that first pools candidate GO terms whose priority keyword (i.e., the keyword most commonly associated with a GO term) appears in the

publication text, then applies BM25 ranking to this filtered candidate set. TF-IDF achieved the highest performance, reaching a Recall@100 of 0.89, indicating that 89% of manually curated GO annotations were recovered within the top 100 predictions.

To better investigate the nature of the GO terms that IDF-based methods fail to capture, we also examined whether 1-step hierarchical neighbours of the manually curated term, which are semantically similar, might be ranked ahead of the curated term. Accordingly, we evaluated the methods using the Hierarchical Recall@K metric. Unlike standard Recall@K, which requires exact GO term matching, Hierarchical Recall@K counts a prediction as correct if either the exact GO term or any of its 1-step hierarchical neighbours (parent or child terms in the GO directed acyclic graph) appears within the top-K predictions. This relaxed criterion accounts for the inherent semantic overlap among closely related GO terms and reflects practical annotation scenarios in which functionally similar terms may be biologically interchangeable. For example, the GO term “vesicle docking involved in exocytosis” (GO:0006904) has its direct child terms (“exocyst assembly”, “synaptic vesicle docking”, etc.) and its parent(s) (“vesicle docking” and “exocytic process”) as its list of 1-step hierarchical neighbours. If the term itself or any of its 1-step hierarchical neighbours appears within the top K predictions, it is counted as correct under the Hierarchical Recall@K metric.

As shown in Figure 7, TF-IDF achieved the highest performance, with a Hierarchical Recall@100 of 93%, compared with 90% for BM25, 83% for IDF-only, and 90% for the Two-Step Ranking method. All methods substantially outperformed the Dummy baseline (Recall@100 = 0.24). The consistent improvement of Hierarchical Recall over exact-match Recall (e.g., TF-IDF: 93% vs. 89% at K=100) demonstrates that a meaningful proportion of missed predictions in the top 100 list correspond to semantically related GO term neighbours rather than random errors. Because the difference between TF-IDF recall and hierarchical recall for the top 100 list is significant, we added

an option in our tool to inject the 1-step hierarchical neighbours of the top 100 GO term candidates into the LLM prompt; however, since this option would considerably increase computational time, the standard setting for the GO-Prioritization module remains restricted to the top 100 candidates.

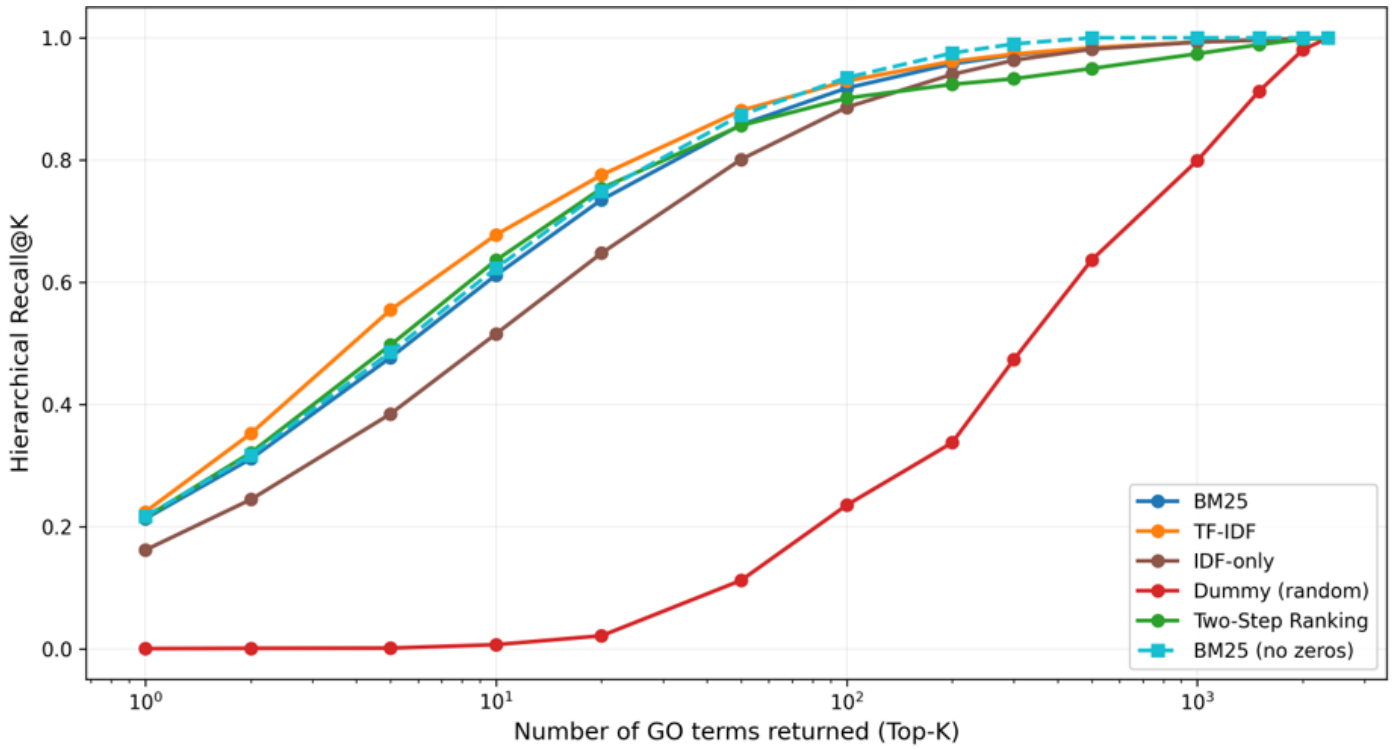

Figure 7. GO-Prioritization Scripts Cross-comparison using Hierarchical Recall@K. Performance of six GO term ranking methods evaluated on 18,890 PMID-ENTITY-GO annotations, with K denoting the number of top-ranked GO terms returned (x-axis, log scale). Hierarchical Recall@K counts a prediction as correct if the exact curated GO term, or one of its direct 1-step hierarchical neighbours in the directed acyclic graph (i.e., a parent or child term), appears among the top-K predictions. Methods compared include BM25 (blue), TF-IDF (orange), IDF-only (brown), Dummy random baseline (dark red), Two-Step Ranking (green), and BM25 excluding zero-score annotations (cyan dashed). TF-IDF showed the strongest performance, reaching a Hierarchical Recall@100 of 0.93, indicating that 93% of curated GO annotations were recovered within the top 100 predictions when direct parent-child GO term relationships were also credited.

IDF-based methods are generally highly efficient, which is the primary reason we chose them for pre-query text mining in our pipeline. TF-IDF had a mean per-annotation runtime of 0.23s, compared with 0.26s for BM25, 0.20s for IDF-only, 0.84s for Two-Step Ranking, and 0.09s for the

Dummy baseline (Figure 8). The Two-Step Ranking method takes significantly longer than the others, likely because it involves an additional step of candidate filtration. Otherwise, given the recall and time results shown in Figures 6-8, the TF-IDF module fulfills its intended purpose in the pipeline by providing a quick, efficient first-pass mechanism that reduces the search space for the LLM by prioritizing high-recall GO terms that are lexically and statistically plausible given the input text.

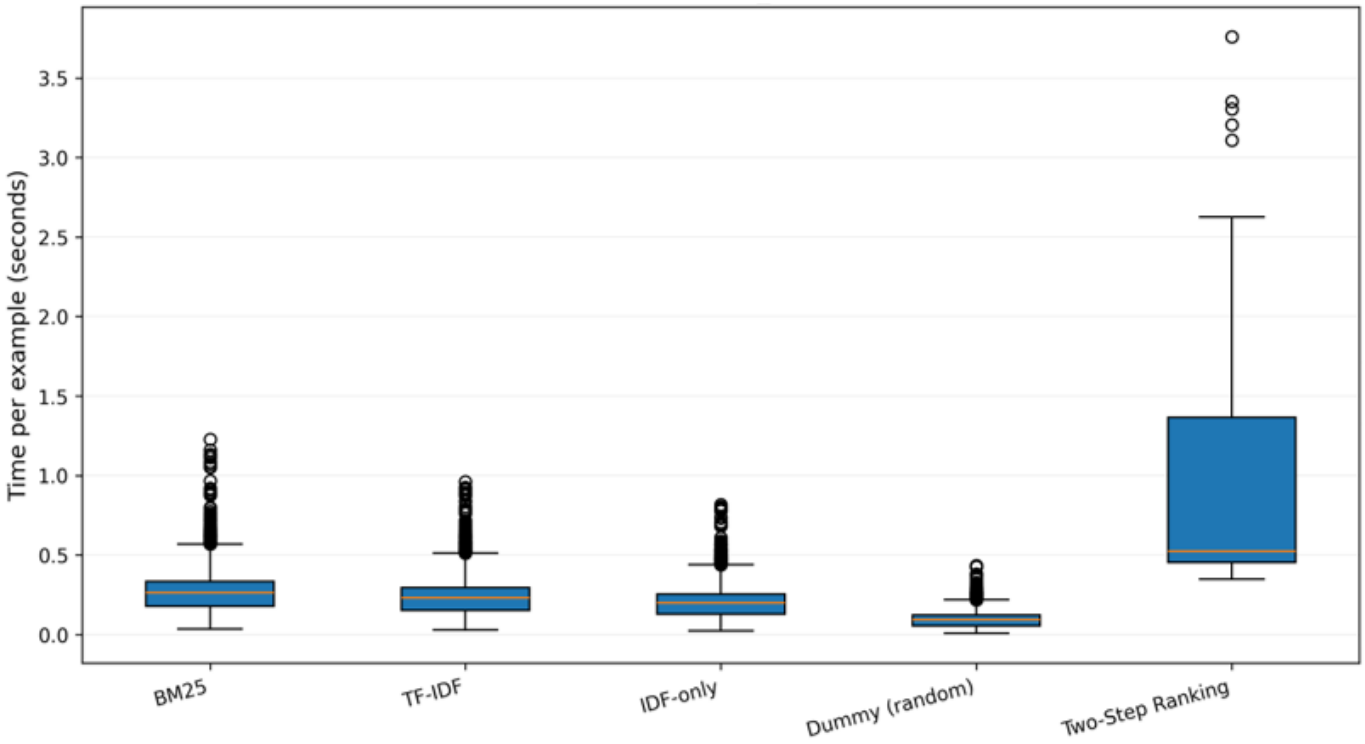

Figure 8. GO-Prioritization Scripts Cross-Comparison (Time). Boxplots show the distribution of per-annotation processing time (seconds) across five ranking methods evaluated on the 18,890 PMID-ENTITY-GO annotations. Boxes represent the interquartile range (IQR); whiskers extend to 1.5×IQR; outliers are shown as individual points. Two-Step Ranking has substantially longer computation times (mean=0.84s) than TF-IDF (mean=0.23s), BM25 (mean=0.26s), IDF-only (mean=0.20s), and the Dummy baseline (mean=0.09s).

3.2 Name-Tracer Module achieves High Recall for Traceable Entities

When tested on 18,890 annotations, the Name-Tracer detects manually curated entities in 15,248 annotations (80.7%) with 100% confidence and in 1,265 additional annotations (6.7%) with ≥ 80% confidence, yielding 87% recall and a mean processing time of 0.14s per query (Figure 9). The main bottleneck of this module is its overreliance on the static, gene-level aliases list. As a result, it misses newly discovered RNA entities and those with non-traditional names. However, it serves as an initial, reasonably accurate filtration script that cues the LLM during the annotation process.

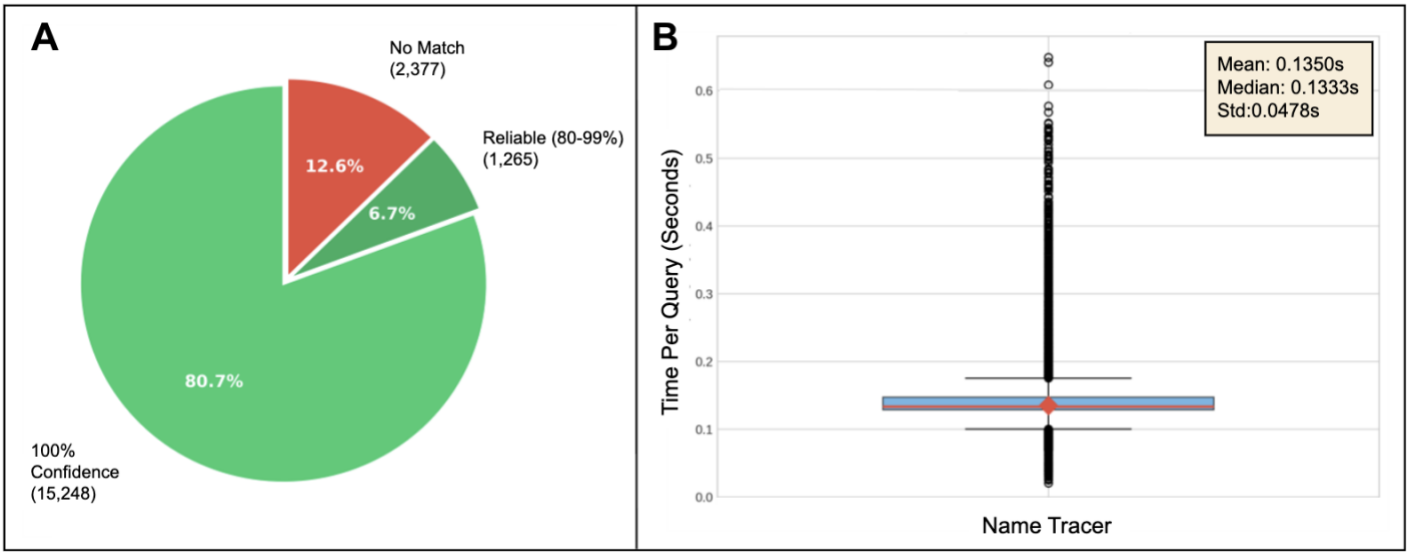

Figure 9. Name-Tracer Module Performance. A. Recall. Recall performance of the Name-Tracer module was evaluated on the 18,890 PMID-ENTITY-GO annotations. The tool traced the manually curated entities in 15,248 annotations (80.7%) with 100% confidence and in 1,265 additional annotations (6.7%) with ≥ 80% confidence, for a total recall of 87%. **B. Annotation Processing Time.** Boxplot showing the distribution of per-annotation processing time (seconds) across the 18,890 annotations. The mean processing time is 0.14s.

3.3 GOFinder-AI achieves High Recall over Gene-Level Annotations

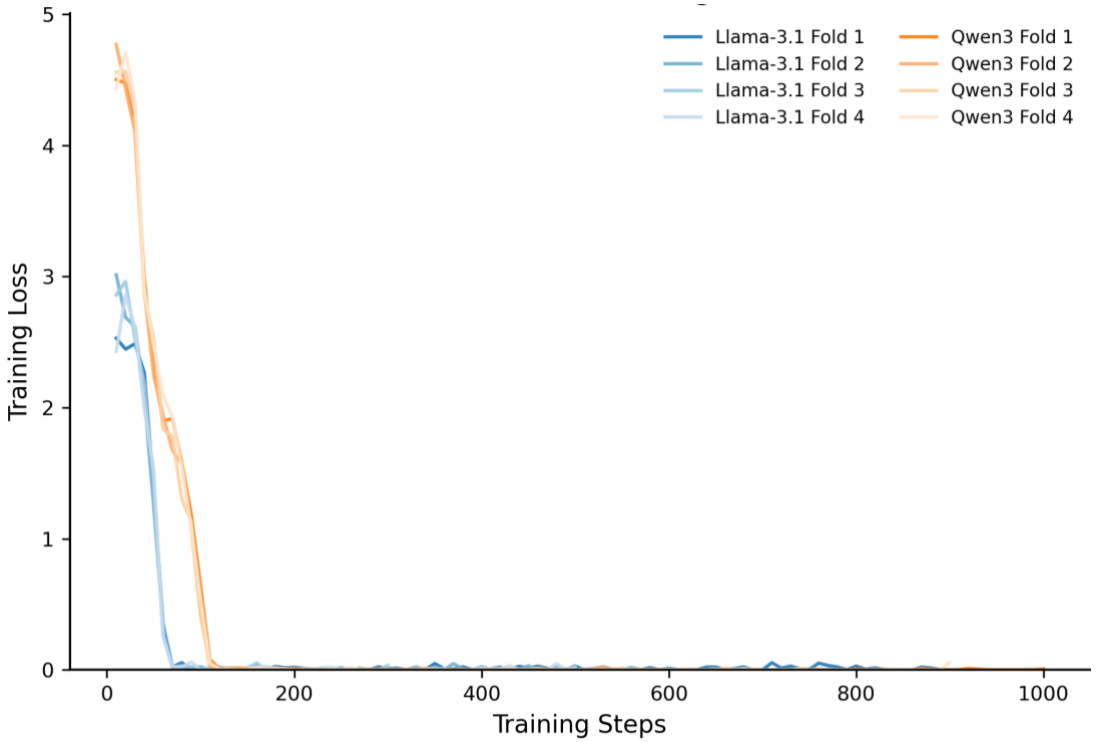

Figure 10. Cross-Validation Training Loss during Supervised Fine-Tuning. Training loss curves for Llama-3.1-8B-Instruct and Qwen3-8B across the four grouped cross-validation folds. In each fold, models were trained on approximately 17,222 annotations. Both models exhibited rapid loss reduction in the early training steps, followed by convergence to near-zero loss.

Our analysis of the cross-validation training loss curves shows that both models learned the GO-term relevance task quickly under the selected QLoRA configuration (Figure 10). The steep early decline in their loss curves indicates rapid adaptation to the classification target, while the later plateau suggests that most of the task-specific gain occurred early in training. The broadly similar convergence pattern across folds supports the stability of the fine-tuning procedure and suggests that performance was not driven by a single favourable partition of the training pool.

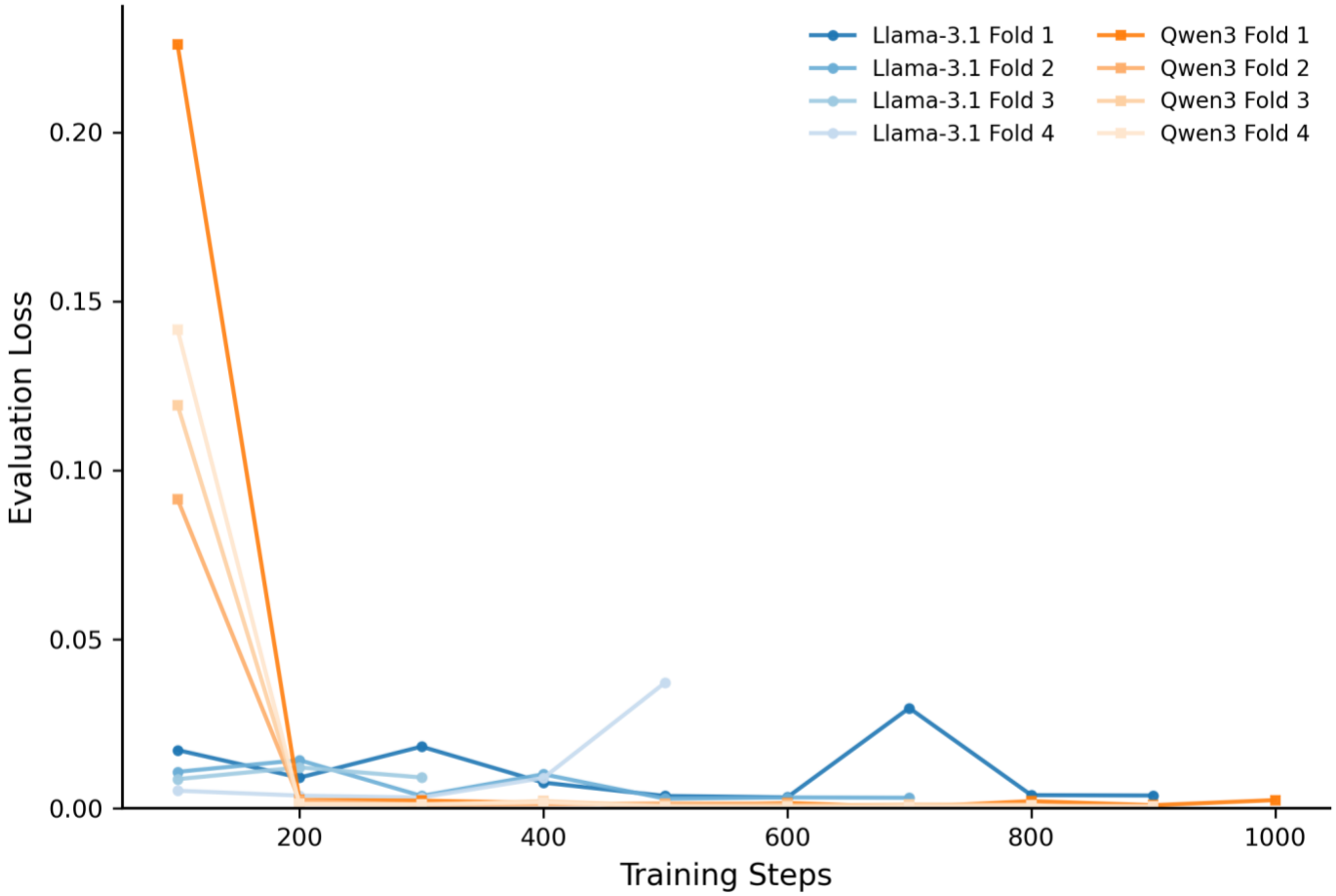

Figure 11. Cross-Validation Evaluation Loss during Supervised Fine-Tuning. Evaluation loss was measured every 100 training steps across four grouped cross-validation runs for Llama-3.1-8B-Instruct and Qwen3-8B. Most runs showed a decreasing or stable evaluation loss over training, with Qwen3-8B generally achieving lower final validation loss than Llama-3.1-8B.

Evaluation loss is more informative than training loss for assessing generalization because it is computed on fold-specific validation data not used for parameter updates. As shown in Figure 11, across all runs, evaluation loss declined and then stabilized without a sustained upward trend, indicating that the selected hyperparameter configurations limited overfitting. Qwen3 achieved a lower final evaluation loss than Llama-3.1 in most folds. The isolated upturn observed in folds 1 and 3 of the Llama model demonstrates the value of early stopping: once validation loss stopped improving, training was halted rather than continuing unnecessarily.

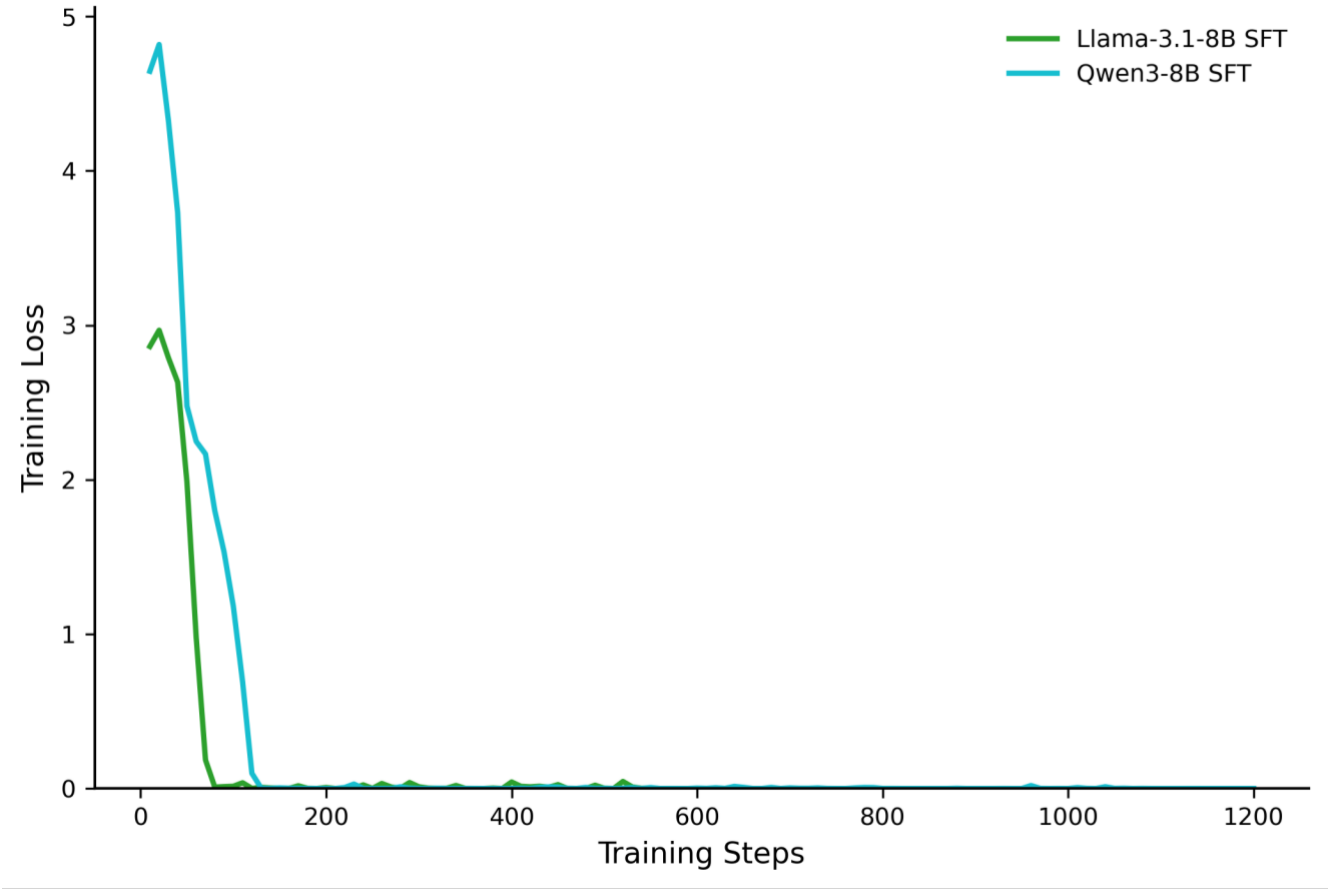

Figure 12. Final Training Loss on the Full Training pool. Training loss curves for the final fine-tuning runs of Llama-3.1-8B and Qwen3-8B on the full 22,964-sample training pool discussed in Section 2.5.

We found that the convergence behaviour observed during cross-validation (Figure 10) was preserved when the models were retrained on the full training pool (Figure 12). Both models again showed rapid early loss reduction, followed by a flattening of the curve, indicating that the selected hyperparameter configurations scaled appropriately from the fold level to final model training.

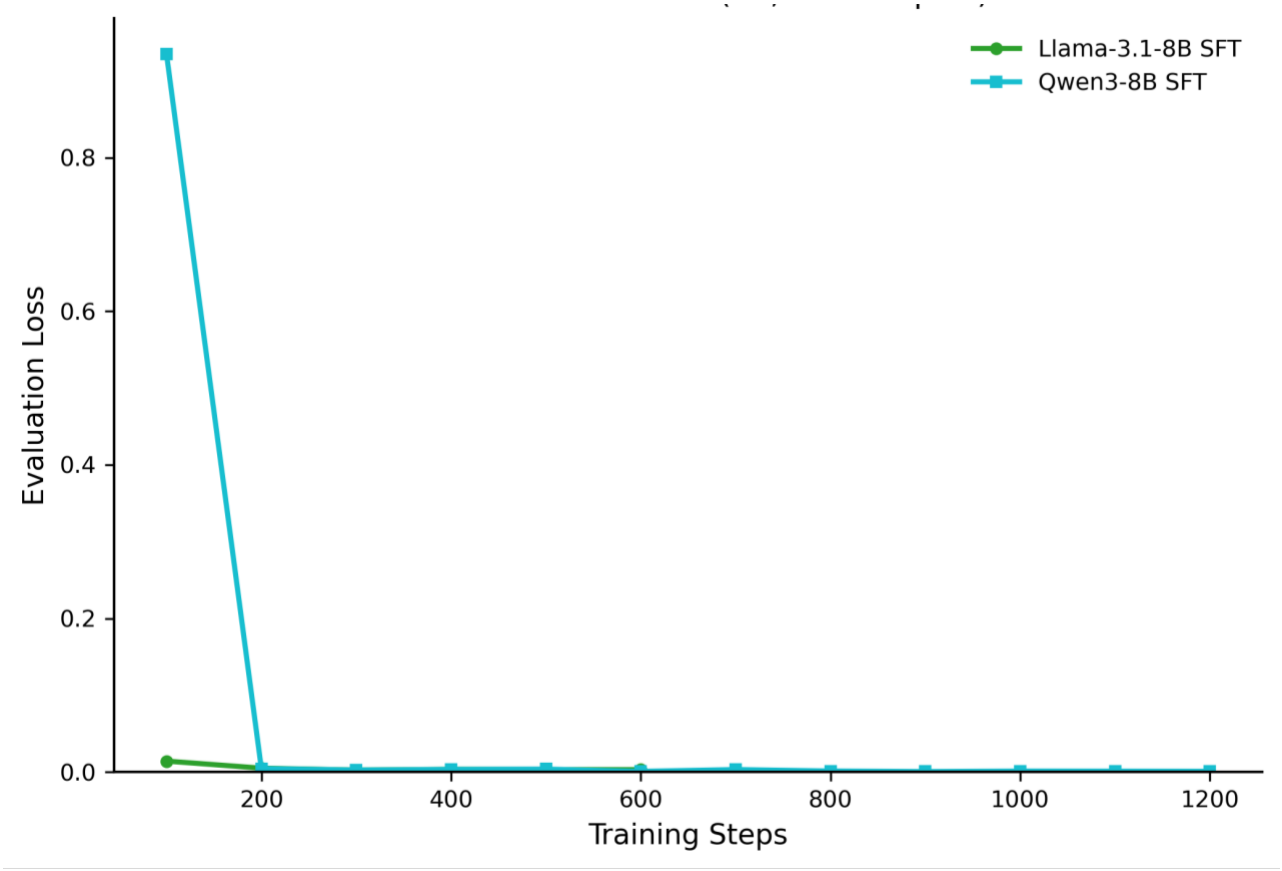

Figure 13. Final Evaluation Loss on Held-Out Validation Data. Evaluation loss during the final fine-tuning of Llama-3.1-8B and Qwen3-8B, measured on held-out validation data used in the final training phase.

Both models also exhibit stable validation behaviour during final training (Figure 13). The absence of a persistent increase in evaluation loss indicates that the models retained generalization while fitting the training data. The combination of limited epochs, gradient clipping, cosine learning-rate decay, and early stopping helped prevent the models from continuing to optimize the training set after validation performance plateaued. The low and stable evaluation losses further suggest that extending training beyond the selected configuration would likely have yielded diminishing returns and led to overfitting.

Model	Accuracy	Precision	Recall	F1
gpt-5-mini	0.8772	1.0000	0.7531	0.8592
Llama-3.1-8B-Instruct	0.8574	0.9372	0.7661	0.8431
Qwen3-8B	0.8077	1.0000	0.6152	0.7617
Finetuned Llama-3.1-8B	**0.9950**	**0.9950**	**0.9950**	**0.9950**
Finetuned Qwen3-8B	**0.9989**	**0.9997**	**0.9981**	**0.9989**

Best zero-shot (gpt-5-mini) Fine-tuned models

Table 1. LLM Performance Cross-Comparison. Classification performance metrics for zero-shot and fine-tuned models, evaluated on the independent test set described in Section 2.10 (n=7,218; 3,609 positive, 3,609 negative examples). Accuracy reflects overall classification correctness. Precision, Recall, and F1 are computed for the TRUE (relevant) class. Fine-tuned models achieved near-perfect performance, with Qwen's and Llama's fine-tuned models attaining accuracies of 99.89% and 99.50%, respectively.

On the independent test set (n=7,218), we find that supervised fine-tuning substantially improved LLM performance on the GO-term relevance classification task compared with relying solely on prompt engineering (Table 1). The largest gain was in recall, indicating that the fine-tuned models were much more reliable at identifying true relevant PMID-GO pairs rather than conservatively returning only high-confidence positives. Both fine-tuned models exceeded 99% accuracy, with the fine-tuned Qwen3-8B model achieving the strongest overall performance. Notably, both fine-tuned models substantially outperformed the proprietary GPT-5 mini model. Overall, these results justify selecting the fine-tuned Qwen3-8B model as the primary classifier in the pipeline.

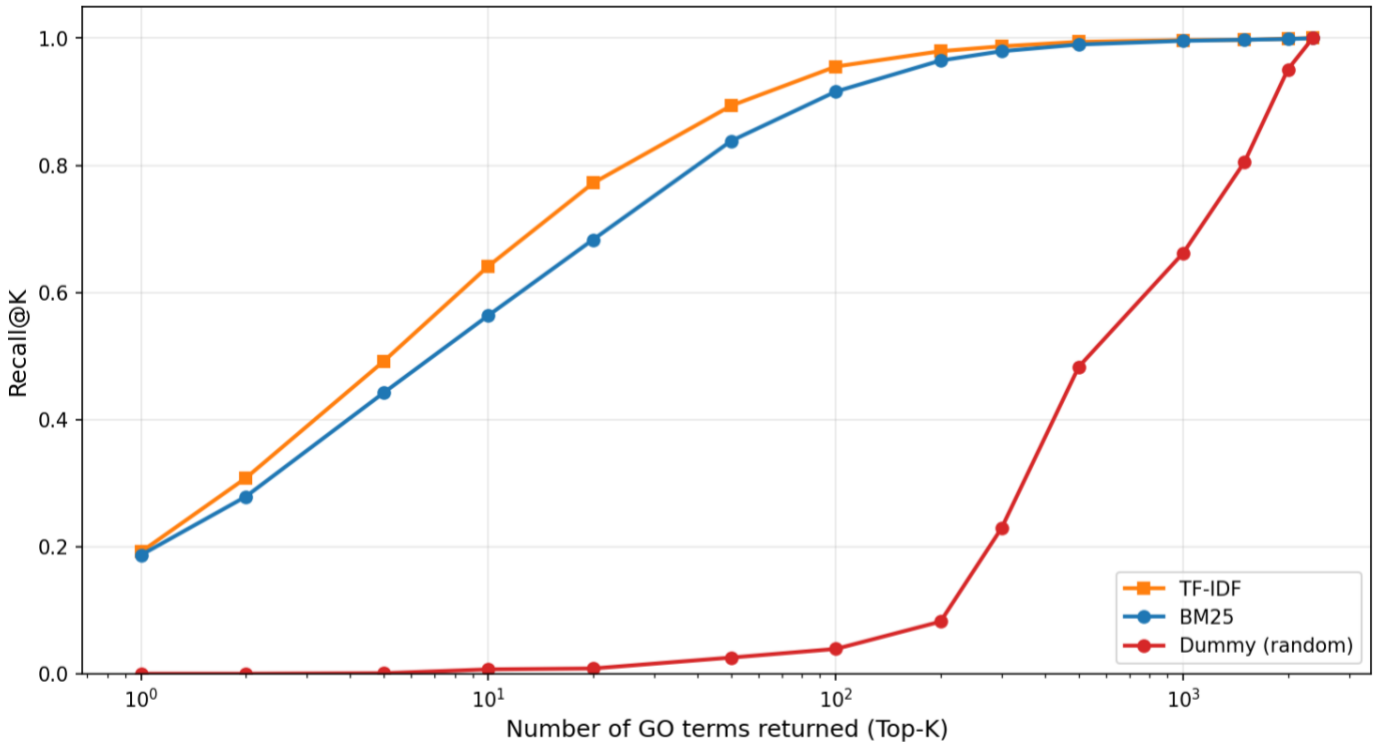

Figure 14. GO Term Prioritization Module Cumulative Accuracy Test. Performance of three GO term ranking methods was evaluated on 3,573 positive annotations from the 7,218-item independent test set, with K representing the number of top-ranked GO terms returned (x-axis, log scale). Methods compared include: BM25 (blue), TF-IDF (orange), and a random dummy baseline (dark red). TF-IDF achieved the highest performance, with a Recall@100 of 0.9554, indicating that 95.5% of manually curated GO annotations were recovered within the top 100 predictions.

In GOFinder-AI, the LLMs do not operate in isolation. The tool uses a two-step process to extract GO annotations from a given text, with text mining preceding the LLM query. To test the impact of adding pre-query text mining on the tool's overall accuracy, we first isolated the 3,573 positive annotations from the 7,218 final, independent set. Then, as shown in Figure 14, we assessed how many of these annotations had their corresponding GO terms recovered among the top 100 candidates recommended by the GO-Prioritization module. The TF-IDF method demonstrates superior performance across all K values, achieving 95.5% recall at K=100. While not perfect, the GO-Prioritization module is crucial. By reducing the candidate space from nearly 40,000 GO terms

to the top 100 ranked terms while retaining 95.5% of relevant annotations, the module substantially lowers the number of LLM evaluations required. This makes the overall pipeline both more efficient and more robust, because the LLM is asked to judge a focused candidate set rather than reason over the entire ontology. On a side note, the semi-logarithmic scale shows that TF-IDF performance plateaus above K=100, indicating diminishing returns from including additional candidates. This supports selecting K=100 as the operational threshold for our tool.

Model	NLP Recall@100	LLM-Only Accuracy	Cumulative Accuracy
GPT-5 mini	0.9554	0.7617	0.7340
Llama-3.1-8B-Instruct	0.9554	0.7778	0.7595
Qwen3-8B-ZeroShot	0.9554	0.6406	0.6304
LLaMA-FineTuned	**0.9554**	**0.9950**	**0.9507**
Qwen3-8B-FineTuned	**0.9554**	**0.9975**	**0.9532**

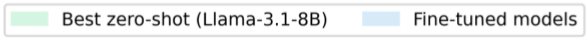

Table 2. End-to-End Cumulative Accuracy of the GOFinder-AI pipeline. Two-stage evaluation results combine the TF-IDF GO-Prioritization module (NLP) with LLM-based relevance classification. Cumulative accuracy reflects the proportion of positive annotations recovered by the retrieval stage and then correctly classified by the LLM. Each component was evaluated separately on the 3,573 positive annotations from the independent 7,218-test set. As reported in Figure 14, the TF-IDF GO-Prioritization module achieved a Recall@100 of 0.9554, indicating that 95.5% of manually curated GO annotations were recovered within the top 100 predictions. Both zero-shot and fine-tuned LLMs were also tested on the positive annotations. The results were then used to estimate the pipeline's end-to-end performance (cumulative accuracy). Qwen's fine-tuned model achieved the highest cumulative accuracy (95.32%), followed closely by Llama's (95.07%).

After measuring GO-Prioritization performance on the held-out positive annotations (Recall@100 = 0.9554), we evaluated the LLMs separately on the same annotation set and then combined the results to estimate the end-to-end performance (cumulative accuracy) of the full GOFinder-AI pipeline, as shown in Table 2. The fine-tuned models remained clearly superior to their zero-shot counterparts, with the fine-tuned Qwen3-8B model achieving the highest cumulative

accuracy (95.32%) and the fine-tuned Llama-3.1-8B model performing similarly (95.07%). The drop from standalone LLM accuracy to cumulative accuracy reflects the cost of retrieval-stage filtering: approximately 4.5% of relevant GO terms were not recovered within the top 100 TF-IDF candidates and were therefore unavailable to the LLM classifier. This finding is important because it shows that GOFinder-AI retains high end-to-end accuracy despite a substantial reduction in the GO search space. In contrast, the zero-shot models were more strongly affected by the two-stage design because their weaker classification performance compounded the recall limits imposed by the retrieval stage.

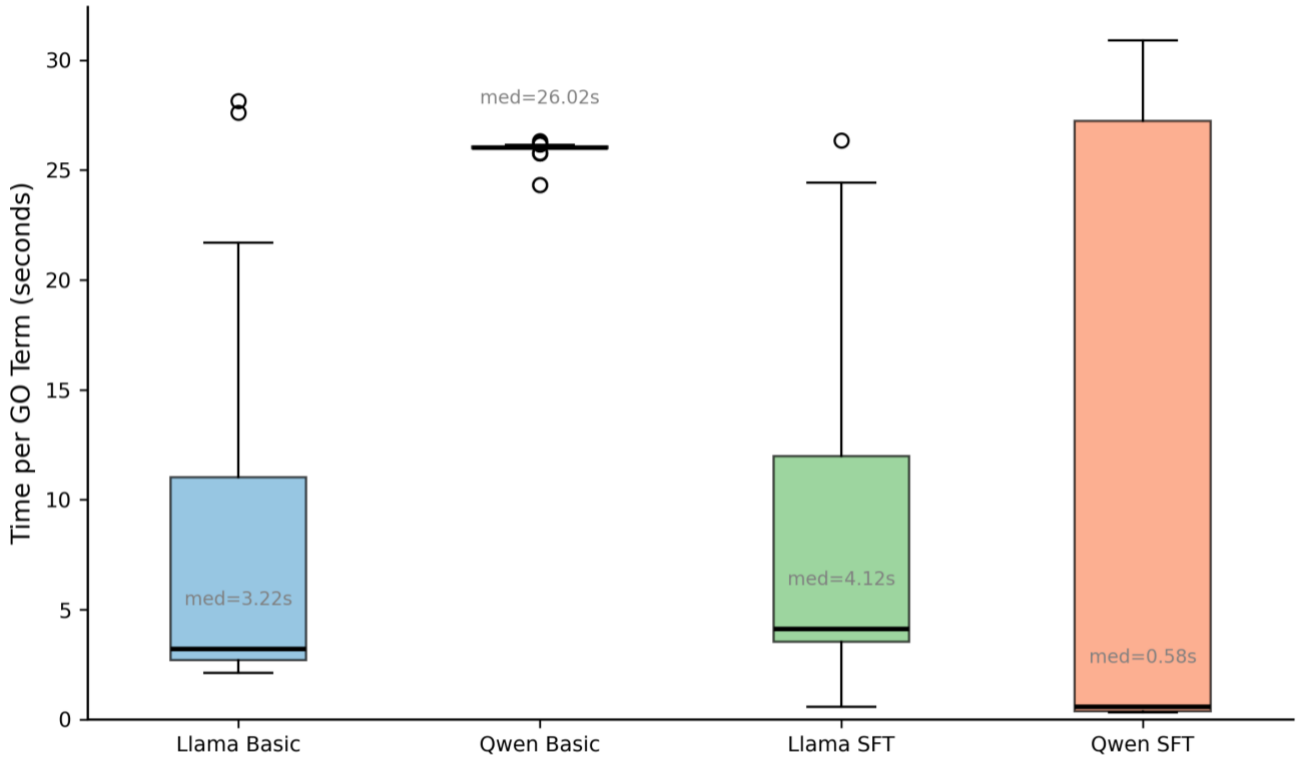

Figure 15. Runtime per GO Term. Boxplots show query execution times per GO term across four model configurations: Llama-3.1-8B zero-shot model (Llama Basic), Qwen3-8B zero-shot model (Qwen Basic), Llama's supervised fine-tuned model (Llama SFT), and Qwen's supervised fine-tuned model (Qwen SFT). The median runtimes were 3.22s, 26.02s, 4.12s, and 0.58s, respectively. Qwen SFT had the lowest median latency.

When assessing the time the LLMs take to process each GO term candidate relayed by the GO-Prioritization module, we found that the fine-tuned Qwen3-8B model is substantially faster than the other models, with a mean processing time of 0.58s per GO term (Figure 15). For instance, at a scale of 100 candidate GO terms, its processing time is roughly 58 seconds, compared with approximately 43 minutes for the zero-shot Qwen baseline. This efficiency likely stems from the compact output format the model learned during fine-tuning, which reduces the need for longer, open-ended generations at inference time. Overall, our cumulative accuracy (Table 2) and time-per-query results (Figure 15) indicate that the fine-tuned Qwen3-8B model is the most favourable LLM for our pipeline.

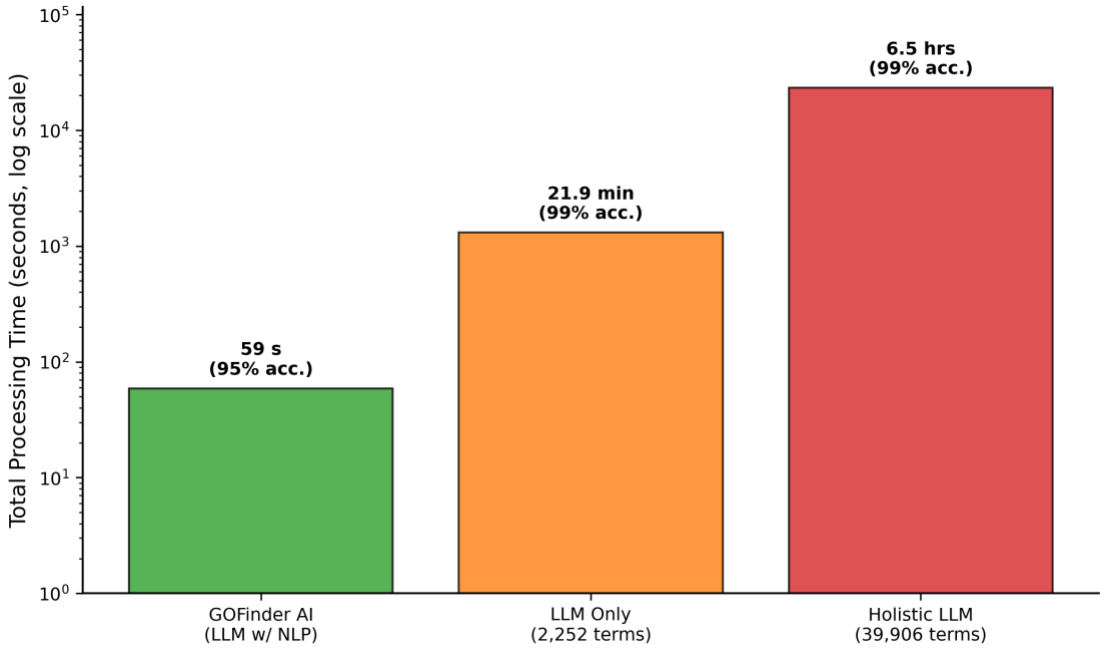

Figure 16. Total Annotation Time for Three GO Annotation Strategies. A log-scale comparison of total processing times for annotating texts averaging 202 words using three strategies: GOFinder-AI with NLP pre-filtering, LLM-only evaluation over the curated 2,252-term GO list, and exhaustive LLM evaluation over all 39,906 GO terms. Mean total processing times were approximately 59 seconds, 21.9 minutes, and 6.5 hours, respectively. As indicated in Table 2, the GOFinder-AI pipeline achieved 95% cumulative accuracy, compared with approximately 99% for the two LLM-only baselines. Processing times were averaged over three separate trials.

By combining pre-query text mining with downstream LLM classification, GOFinder-AI takes approximately 59 seconds per document while maintaining 95% cumulative accuracy (Figure 16). In contrast, applying the LLM directly to the curated 2,252-term GO list required approximately 21.9 minutes per document, and exhaustive evaluation across all 39,906 GO terms required approximately 6.5 hours. This comparison shows that retrieval-first filtering makes routine GO annotation feasible without degrading performance. The reduction in cumulative accuracy from 99% to 95% is the principal cost of search-space reduction, but we believe this trade-off is reasonable given the substantial improvement in throughput.

3.4 GOFinder-AI User-Interactive Site Dockerizes All Parts of the Pipeline

In our React-based web interface, we integrated all components of the GOFinder-AI pipeline across the modalities shown in Figure 17: (1) GO Light, our primary modality, traces relevant GO annotations and their corresponding entities from any text. (2) GO Relevance determines whether a particular GO term is relevant to an entity given a text. (3) GO Extract extracts GO term annotations without relying on our pre-query text-mining scripts, serving as a quality control for our GO-Prioritization and Name-Tracer modules. (4) GO Automate connects our pipeline to NCBI-PubMed via PubMed Extract. This modality automates curation for any provided list of entities. Our interface handles both typed and PDF text. It also includes a Save/Discard framework that lets the user choose which annotations to keep and provides feedback to the LLM for subsequent reinforcement learning.

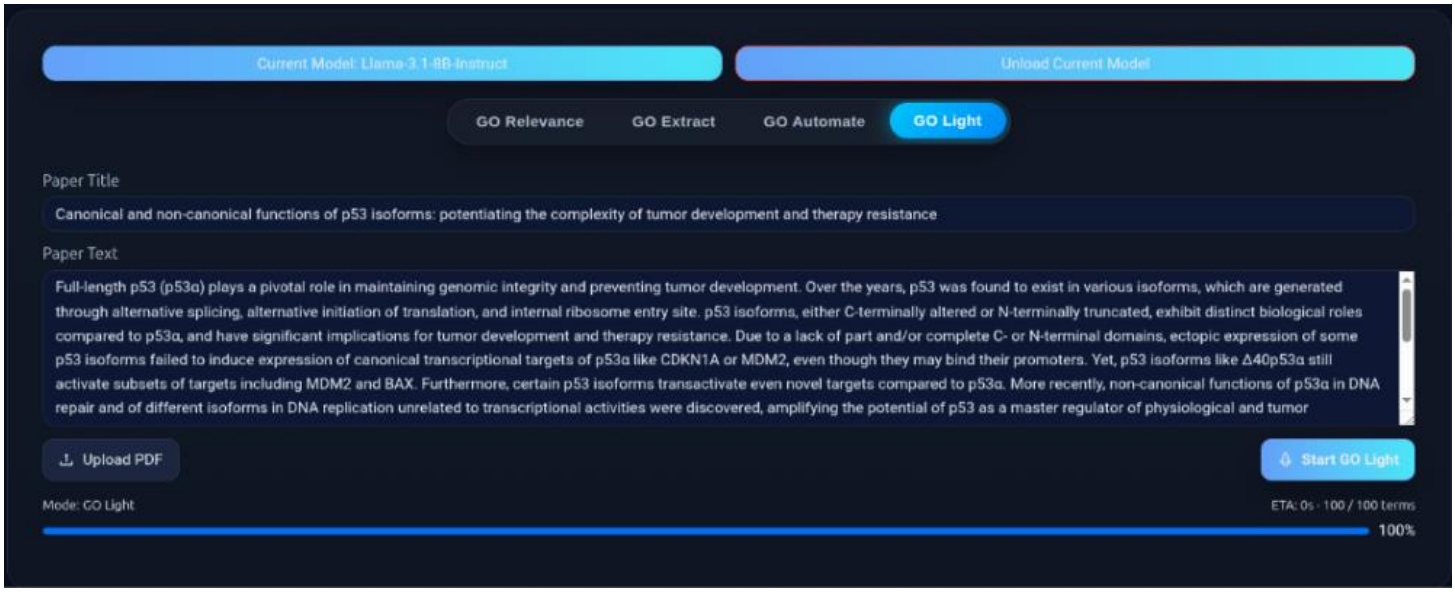

Figure 17. GOFinder-AI User Interface. Pre-Query Interface. This panel shows the initial setup of the GOFinder-AI tool, in which the entire workflow described in Figure 5 is automated for user-provided text (typed or from a PDF). Users load their preferred LLM (e.g., fine-tuned Qwen-3.1-8B), provide the paper title and text, then click the start button to begin the analysis.

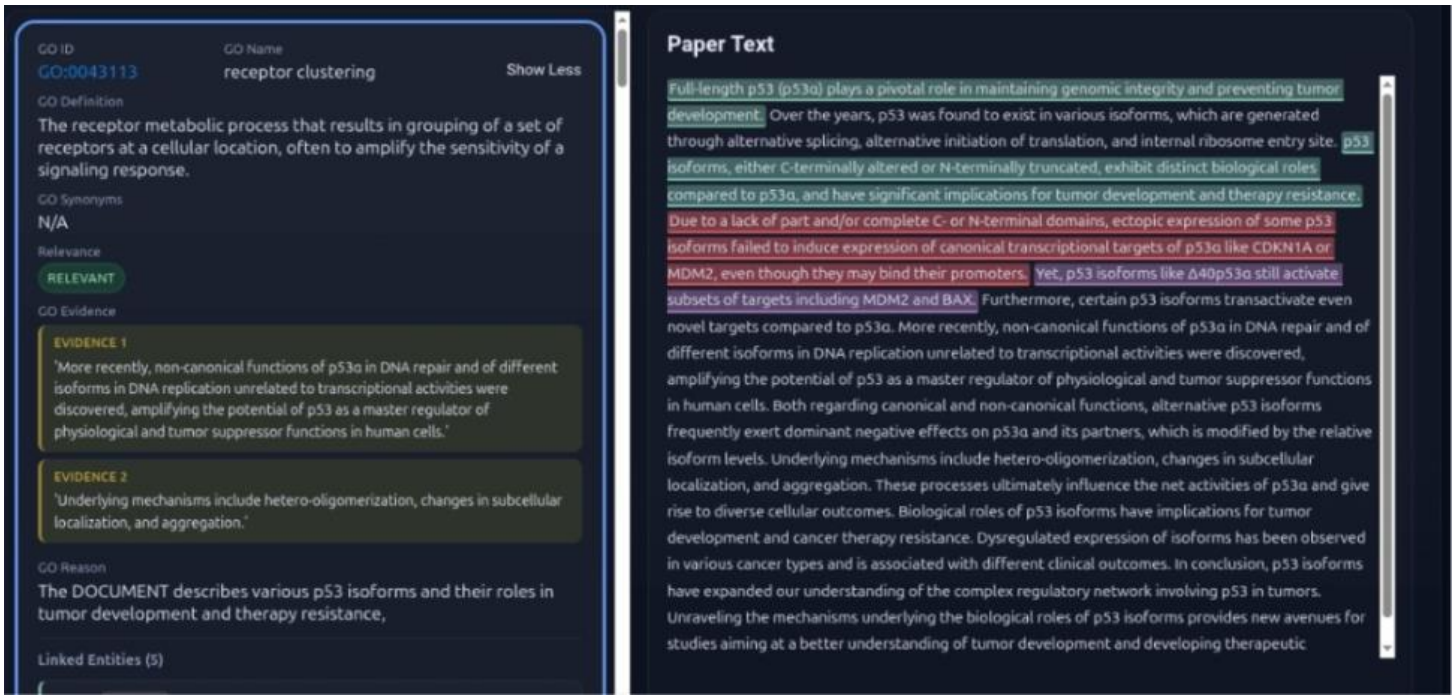

Figure 18. GOFinder-AI User Interface. Post-Query. This panel displays the results interface after the query shown in Figure 17 is executed**.** 3 extracted excerpts are highlighted alongside the tool's corresponding relevant GO term, “receptor clustering”.

3.5 GOFinder-AI Traces Novel Annotations

To test GOFinder-AI's potential to enrich poorly annotated genes with novel GO annotations, we isolated a set of 30 hallmark genes that are among the least annotated in AmiGO. We then connected our GOFinder-AI pipeline to PubMed Extract, a rule-based retrieval pipeline we developed to identify relevant publications for any given biological entity in the NCBI PubMed database, via the "GO Automate" modality. PubMed Extract retrieved the most relevant publications for each of the 30 hallmark genes. We extracted the full text of these publications and fed them into the tool. The pipeline validated existing annotations with recent citations and, more importantly, retrieved novel GO annotations from recent publications.

For instance, KLHDC8A (Kelch Domain Containing 8A) is one of the hallmark genes we isolated. It plays a crucial role in glioblastoma biology, and higher KLHDC8A expression has been associated with tumour aggressiveness [74]. Mechanistically, KLHDC8A has been reported to promote glioma cell proliferation, migration, and invasion [74]. KLHDC8A also promotes primary ciliogenesis and Hedgehog signalling, supporting tumour growth and self-renewal [75]. Despite these well-supported roles, the gene is annotated to only one GO term in the AmiGO database: protein binding (GO:0005515). Using our pipeline, we validated the protein binding annotation and retrieved evidence-based annotations for macrophage activation (GO:0042116), regulation of cilium assembly (GO:1902017), and Wnt signalling pathway (GO:0016055), among others. Another example is YPEL1 (Protein yippee-like 1). It is a hallmark gene long associated with the regulation of cell morphology and developmental organization, and more recent work suggests that it may suppress glioma proliferation and invasion [76,77]. The terms annotated for YPEL1 in the AmiGO database are nucleus (GO:0005634) and metal ion binding (GO:0046872), both of which are considerably generic and not reflective of the gene's literature-supported functional roles. Using GOFinder-AI, we

validated both annotations and retrieved evidence-based annotations for regulation of cell cycle (GO:0051726), cellular senescence (GO:0090398), and mesenchymal-to-epithelial transition (GO:0060231), among others. In both cases, our pipeline provided deeper insight into two hallmark genes that are otherwise missing from the AmiGO database.

Hallmark Gene	Novel GO Annotation	PMID
EPSTI1	GO:0019882	PMID: 40459882
EPSTI1	GO:0070098	PMID: 40459882
EPSTI1	GO:0140888	PMID: 40459882
EPSTI1	GO:0070498	PMID: 40459882
EPSTI1	GO:0038061	PMID: 40459882
EPSTI1	GO:0045670	PMID: 39083969
EPSTI1	GO:0045124	PMID: 39083969
EPSTI1	GO:0030155	PMID: 35778487
YPEL1	GO:0051726	PMID: 40103457
YPEL1	GO:0090398	PMID: 40103457
YPEL1	GO:0032502	PMID: 40103457
YPEL1	GO:0060231	PMID: 40103457
YPEL1	GO:0060429	PMID: 40103457
YPEL1	GO:0009888	PMID: 40103457
YPEL1	GO:0000902	PMID: 11473580
YPEL1	GO:0030855	PMID: 11473580
KLHDC8A	GO:0042116	PMID:39327249
KLHDC8A	GO:0007224	PMID: 36394953
KLHDC8A	GO:1902017	PMID: 36394953
KLHDC8A	GO:0008283	PMID: 37814858

KLHDC8A	GO:0016477	PMID: 37814858
KLHDC8A	GO:0010468	PMID: 32851798
KLHDC8A	GO:0000165	PMID: 32851798
KLHDC8A	GO:0038066	PMID: 32851798
KLHDC8A	GO:0016477	PMID: 32851798
KLHDC8A	GO:0016055	PMID: 39614999
FBXO34	GO:1902749	PMID: 33842473
FBXO34	GO:0001556	PMID: 33842473
FBXO34	GO:0019042	PMID 36285453
FBXO34	GO:0016567	PMID 36285453

Table 3. Novelty List Table. List of 30 novel GO annotations identified by GOFinder-AI for four target genes (EPSTI1, YPEL1, KLHDC8A, and FBXO34) using the GO Automate modality. For each annotation, the pipeline retrieves a supporting PMID (shown in the third column) and provides citation-based evidence and reasoning to support the predicted gene–GO association (not shown in the table).

The cases of KLHDC8A, YPEL1, and the other genes examined in Table 3 support the observation that substantial functional evidence remains in the text and is not captured in computable GO annotations, primarily because manual curation lags behind publication rates. This demonstrates GOFinder-AI's potential to address the gap in poorly annotated genes by leveraging the available supporting literature.

4. Discussion

In this work, we aimed to address a central challenge in functional genomics: how to convert the rapidly expanding biomedical literature into reliable, evidence-linked GO annotations at a scale that manual curation alone cannot sustain. As discussed, existing automated approaches are effective at propagating known functional signals but are less suited to extracting and justifying GO annotations directly from newly published text. At the same time, LLMs offer clear potential for literature-grounded annotation, yet their outputs must be constrained, validated, and made interpretable to be useful in scientific workflows. GOFinder-AI was developed in response to this need. More specifically, we tested the hypothesis that a retrieval-first, fine-tuned LLM pipeline could accurately extract literature-grounded GO annotations from biomedical text, outperform baseline text-mining and zero-shot LLM approaches, and provide a practical framework for the functional annotation of poorly characterized human genes.

The results support this hypothesis and meet our three objectives. The GO-Prioritization module achieved high recovery of relevant GO terms within a reduced candidate space; the Name-Tracer module achieved high recall for traceable entities; and the fine-tuned LLMs substantially outperformed their zero-shot counterparts in classifying GO-term relevance. When integrated into the full pipeline, GOFinder-AI maintained high cumulative accuracy while processing text efficiently. More notably, we demonstrated, with 30 novel annotations, that GOFinder-AI could generate evidence-linked annotations for poorly annotated genes.

4.1 Significance

One important contribution of GOFinder-AI is its value as a practical bioinformatics research tool. Functional annotation remains a central challenge in genomics and molecular biology, particularly for genes and other biological entities that are poorly characterized in existing databases. By combining pre-query text mining, entity tracing, and LLM-based relevance classification, GOFinder-AI enables researchers to generate up-to-date, evidence-based candidate annotations directly from the biomedical literature. This is especially useful when researchers aim to investigate the potential functions of a biological entity using recent studies rather than relying solely on static database entries. In this sense, GOFinder-AI serves not only as an annotation system but also as a literature-grounded hypothesis-support tool that helps researchers identify functionally relevant GO terms, supporting citations, and associated entities in a more efficient and interpretable manner.

A second major contribution of GOFinder-AI is its role as an automated curation framework. The results of this thesis demonstrate that combining pre-query text mining with fine-tuned open-weight LLMs can produce a scalable, accurate, and comparatively transparent pipeline for GO annotation from text. This is important because manual curation, while rigorous, continues to lag behind the rate at which new biomedical literature is produced. GOFinder-AI addresses this gap by providing a structured workflow that first reduces the GO search space, then evaluates candidate terms with a task-specific LLM classifier, and finally generates supporting citations and reasoning for the proposed annotations. In doing so, the tool offers a practical intermediate model between fully manual curation and broad automatic annotation. Rather than replacing human expertise, the tool is better understood as a curation-assistance framework that can help accelerate literature-grounded GO annotation and support future efforts to address gaps in the GO database.

Finally, GOFinder-AI does not function as a black-box label generator. Instead, it produces candidate annotations with supporting text, linked entities, and explanatory reasoning, enabling users to inspect the basis of each prediction before deciding whether to retain or reject it. This interpretability is essential for the adoption of LLM-based annotation tools in scientific environments, where evidence traceability is as important as predictive performance. In this sense, GOFinder-AI contributes both a technical workflow and a practical model for designing GO curation systems responsibly.

4.2 Limitations and Future Work

Despite its promising performance, GOFinder-AI has several limitations that also point to key directions for its future development. The first limitation concerns the nature of the AmiGO-sourced datasets used for supervised fine-tuning. Although these datasets provided a valuable foundation for training the pipeline's relevance-classification component, the annotations they contain generally lack explicit supporting citation spans or human-readable reasoning for each annotation. As a result, they were best suited to fine-tuning LLMs on a narrower classification task: determining whether a candidate GO term is IS_RELEVANT or NOT_RELEVANT to a given input text. We aim to address this limitation in two ways. First, the current GOFinder-AI pipeline already generates annotations with supporting citations, reasoning, and linked entities. Once these outputs are internally validated, they may form the basis of richer future training datasets that support direct fine-tuning on additional tasks such as citation extraction and explanation generation. Second, as part of the broader development of GOFinder-AI, we manually curated a set of 1,086 isoform-level GO annotations, each including supporting citations and reasoning. Although isoform-level annotation was not the primary focus of

the present thesis, this dataset represents an important resource for future extensions to train and evaluate models on higher-resolution functional annotation tasks, with an emphasis on high-quality evidence and reasoning.

A second limitation is that the current pipeline does not fully address the challenge of assigning the most appropriate GO evidence code and the associated biological evidence context to each generated annotation. In the GO framework, evidence codes describe the basis for an annotation. For example, experimentally grounded codes such as IDA or IMP carry a different meaning from literature-based codes such as TAS, in which the annotation is supported by an explicit statement in the source text. Because GOFinder-AI is designed to extract annotations from textual evidence, the default evidence code most closely aligned with its outputs is TAS. This allows the pipeline to rapidly convert explicit statements in the literature into computationally useful candidate annotations. However, if such annotations were incorporated into AmiGO or related GO resources, the biological model system and the exact evidence type would still require manual inspection. In other words, the current system can support annotation generation, but it does not yet fully automate curator-level evidence-code assignment.

A third limitation stems from the pipeline's retrieval-first design. The text-mining layer that reduces the GO search space is a major strength of GOFinder-AI, making the annotation process more efficient and interpretable. However, by narrowing the candidate space before LLM evaluation, the system may overlook potentially relevant GO terms that were not ranked highly enough in the retrieval stage. This introduces a trade-off between efficiency and completeness. One future direction would be to strengthen the retrieval layer by incorporating more advanced retrieval approaches beyond TF-IDF, including systems that better capture semantic similarity rather than relying primarily on lexical overlap, such as BioBERT [26]. Another possible extension would be to combine the

current GO Light workflow with a limited number of candidate annotations proposed by the more open-ended GO Extract mode. In this design, a small number of extraction-based candidates could be added to the ranked retrieval set before final evaluation. Such a hybrid strategy could improve recall by recovering GO terms missed during prioritization, although it would also increase the risk of hallucinated or weakly supported annotations. Future work should therefore examine whether the gain in coverage justifies the added noise.

Taken together, these limitations do not undermine the value of GOFinder-AI; rather, they mark the next stage of its development. The present work establishes that a retrieval-first, fine-tuned LLM pipeline can support accurate and efficient GO annotation from text and produce evidence-linked outputs that are meaningful for downstream biological interpretation. Future work can now focus on expanding the range of tasks the model is trained to perform, improving evidence-code specificity, and strengthening retrieval to preserve efficiency gains without excluding biologically meaningful annotation candidates.

References

[1] Qiu, S., Yu, G., Lu, X., Domeniconi, C., & Guo, M. (2022). Isoform function prediction by Gene Ontology embedding. *Bioinformatics, 38(19), 4581–4588.* *https://doi.org/10.1093/bioinformatics/btac576*

[2] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., & Sherlock, G. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics, 25(1), 25–29.* *https://doi.org/10.1038/75556*

[3] Gene Ontology Consortium. (2026). The Gene Ontology knowledgebase in 2026. *Nucleic Acids Research, 54(D1), D1779–D1792.* *https://doi.org/10.1093/nar/gkaf1292*

[4] Ersoz, N. S., Bakir-Gungor, B., & Yousef, M. (2023). GeNetOntology: Identifying affected gene ontology terms via grouping, scoring, and modeling of gene expression data utilizing biological knowledge-based machine learning. Frontiers in Genetics, 14, 1139082. https://doi.org/10.3389/fgene.2023.1139082

[5] Ma, T., & Zhang, A. (2019). Incorporating biological knowledge with factor graph neural network for interpretable deep learning. arXiv. https://doi.org/10.48550/arXiv.1906.00537

[6] Huntley, R. P., Sawford, T., Mutowo-Meullenet, P., Shypitsyna, A., Bonilla, C., Martin, M. J., & O'Donovan, C. (2015). The GOA database: Gene Ontology annotation updates for 2015. *Nucleic Acids Research, 43(D1), D1057–D1063.* *https://doi.org/10.1093/nar/gku1113*

[7] Balakrishnan, R., Harris, M. A., Huntley, R., Van Auken, K., & Cherry, J. M. (2013). A guide to best practices for Gene Ontology (GO) manual annotation. Database, 2013, bat054. https://doi.org/10.1093/database/bat054

[8] Jin, Q., Leaman, R., & Lu, Z. (2024). PubMed and beyond: biomedical literature search in the age of artificial intelligence. EBioMedicine, 100, 104988. https://doi.org/10.1016/j.ebiom.2024.104988

[9] Sayers, E. W., Beck, J., Bolton, E. E., Bourexis, D., Brister, J. R., Canese, K., Comeau, D. C., Funk, K., Kim, S., Klimke, W., Marchler-Bauer, A., Landrum, M., Lathrop, S., Lu, Z., Madden, T. L., O'Leary, N., Phan, L., Rangwala, S. H., Schneider, V. A., Skripchenko, Y., … Sherry, S. T. (2021). Database resources of the National Center for Biotechnology Information. Nucleic Acids Research, 49(D1), D10–D17. https://doi.org/10.1093/nar/gkaa892

[10] Baumgartner, W. A., Jr., Cohen, K. B., Fox, L. M., Acquaah-Mensah, G., & Hunter, L. (2007). Manual curation is not sufficient for annotation of genomic databases. *Bioinformatics, 23(13), i41–i48.* *https://doi.org/10.1093/bioinformatics/btm229*

[11] Škunca, N., Altenhoff, A., & Dessimoz, C. (2012). Quality of computationally inferred Gene Ontology annotations. *PLoS Computational Biology, 8(5), e1002533.* *https://doi.org/10.1371/journal.pcbi.1002533*

[12] Burge, S., Kelly, E., Lonsdale, D., Mutowo-Muellenet, P., McAnulla, C., Mitchell, A., Sangrador-Vegas, A., Yong, S.-Y., Mulder, N., & Hunter, S. (2012). Manual GO annotation of predictive protein signatures: The InterPro approach to GO curation. *Database, 2012, bar068.* *https://doi.org/10.1093/database/bar068*

[13] Gaudet, P., Livstone, M. S., Lewis, S. E., & Thomas, P. D. (2011). Phylogenetic-based propagation of functional annotations within the Gene Ontology consortium. *Briefings in Bioinformatics, 12(5), 449–462.* *https://doi.org/10.1093/bib/bbr042*

[14] Finn, R. D., Attwood, T. K., Babbitt, P. C., Bateman, A., Bork, P., Bridge, A. J., Chang, H.-Y., Dosztányi, Z., El-Gebali, S., Fraser, M., Gough, J., Haft, D., Holliday, G. L., Huang, H., Huang, X., Letunic, I., Lopez, R., Lu, S., Marchler-Bauer, A., … Mitchell, A. L. (2017). InterPro in 2017—Beyond protein family and domain annotations. Nucleic Acids Research, 45(D1), D190–D199. https://doi.org/10.1093/nar/gkw1107

[15] UniProt Consortium. (2023). UniProt: The universal protein knowledgebase in 2023. Nucleic Acids Research, 51(D1), D523–D531. https://doi.org/10.1093/nar/gkac1052

[16] MacDougall, A., Volynkin, V., Saidi, R., Poggioli, D., Zellner, H., Hatton-Ellis, E., Joshi, V., O'Donovan, C., Huntley, R. P., & Martin, M. J. (2020). UniRule: A unified rule resource for automatic annotation in the UniProt Knowledgebase. Bioinformatics, 36(17), 4643–4648. https://doi.org/10.1093/bioinformatics/btaa485

[17] Schnoes, A. M., Brown, S. D., Dodevski, I., & Babbitt, P. C. (2009). Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies. PLoS Computational Biology, 5(12), e1000605. https://doi.org/10.1371/journal.pcbi.1000605

[18] Jones, C. E., Brown, A. L., & Baumann, U. (2007). Estimating the annotation error rate of curated GO database sequence annotations. BMC Bioinformatics, 8, 170. https://doi.org/10.1186/1471-2105-8-170

[19] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24(5), 513–523.* *https://doi.org/10.1016/0306-4573(88)90021-0*

[20] Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28(1), 11–21.* *https://doi.org/10.1108/eb026526*

[21] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval, 3(4), 333–389.* *https://doi.org/10.1561/1500000019*

[22] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[23] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[24] Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., & Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. Bioinformatics, 33(14), i37–i48. https://doi.org/10.1093/bioinformatics/btx228

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

[26] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682

[27] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners. arXiv. https://doi.org/10.48550/arXiv.2005.14165

[28] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv. https://doi.org/10.48550/arXiv.2203.02155

[29] Dada, A., Koras, O. A., Bauer, M., Butler, A., Smith, K. E., Kleesiek, J., & Friedrich, J. (2025). MeDiSumQA: Patient-oriented question-answer generation from discharge letters. In S. Ananiadou, D. Demner-Fushman, D. Gupta, & P. Thompson (Eds.), Proceedings of the Second Workshop on Patient-Oriented Language Processing (CL4Health) (pp. 124–136). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.cl4health-1.10

[30] Lin, K.-H., Kao, T.-H., Wang, L.-C., Kuo, C.-T., Chen, P. C.-H., Chu, Y.-C., & Yeh, Y.-C. (2025). Benchmarking large language models GPT-4o, Llama 3.1, and Qwen 2.5 for cancer genetic variant classification. npj Precision Oncology, 9(1), 141. https://doi.org/10.1038/s41698-025-00935-4

[31] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), Article 248. https://doi.org/10.1145/3571730

[32] Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., & Hajishirzi, H. (2023). When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 9802–9822). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.546

[33] Liu, S., McCoy, A. B., & Wright, A. (2025). Improving large language model applications in biomedicine with retrieval-augmented generation: A systematic review, meta-analysis, and clinical development guidelines. *Journal of the American Medical Informatics Association, 32(4), 605–615. https://doi.org/10.1093/jamia/ocaf008*

[34] Jeong, M., Sohn, J., Sung, M., & Kang, J. (2024). Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. Bioinformatics, 40(Suppl 1), i119–i129. https://doi.org/10.1093/bioinformatics/btae238

[35] Alkhalaf, M., Yu, P., Yin, M., & Deng, C. (2024). Applying generative AI with retrieval augmented generation to summarize and extract key clinical information from electronic health records. Journal of Biomedical Informatics, 156, 104662. https://doi.org/10.1016/j.jbi.2024.104662

[36] 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., … Ma, Z. (2024). The Llama 3 herd of models. arXiv. https://doi.org/10.48550/arXiv.2407.21783

[37] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., … Qiu, Z. (2025). Qwen3 technical report. arXiv. https://doi.org/10.48550/arXiv.2505.09388

[38] Sivarajkumar, S., Kelley, M., Samolyk-Mazzanti, A., Visweswaran, S., & Wang, Y. (2024). An empirical evaluation of prompting strategies for large language models in zero-shot clinical natural language processing: Algorithm development and validation study. *JMIR Medical Informatics, 12, e55318. https://doi.org/10.2196/55318*

[39] Tran, H., Yang, Z., Yao, Z., & Yu, H. (2024). BioInstruct: Instruction tuning of large language models for biomedical natural language processing. *Journal of the American Medical Informatics Association, 31(9), 1821–1832. https://doi.org/10.1093/jamia/ocae122*

[40] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. arXiv. https://doi.org/10.48550/arXiv.2106.09685

[41] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. arXiv. https://doi.org/10.48550/arXiv.2305.14314

[42] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. arXiv. https://doi.org/10.48550/arXiv.2305.18290

[43] Lin, Y., Lin, H., Xiong, W., Diao, S., Liu, J., Zhang, J., Pan, R., Wang, H., Hu, W., Zhang, H., Dong, H., Pi, R., Zhao, H., Jiang, N., Ji, H., Yao, Y., & Zhang, T. (2024). Mitigating the alignment tax of RLHF. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language

Processing (pp. 580–606). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.emnlp-main.35

[44] González Barman, K., Lohse, S., & de Regt, H. W. (2025). Reinforcement learning from human feedback in LLMs: Whose culture, whose values, whose perspectives? Philosophy & Technology, 38, 35. https://doi.org/10.1007/s13347-025-00861-0

[45] Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., & Lewis, S. (2009). AmiGO: Online access to ontology and annotation data. Bioinformatics, 25(2), 288–289. https://doi.org/10.1093/bioinformatics/btn615

[46] Binns, D., Dimmer, E., Huntley, R., Barrell, D., O'Donovan, C., & Apweiler, R. (2009). QuickGO: A web-based tool for Gene Ontology searching. Bioinformatics, 25(22), 3045–3046. https://doi.org/10.1093/bioinformatics/btp536

[47] Zhou, N., Jiang, Y., Bergquist, T. R., Lee, A. J., Kacsoh, B. Z., Crocker, A. W., Lewis, K. A., Georghiou, G., Nguyen, H. N., Hamid, M. N., Davis, L., Dogan, T., Atalay, V., Rifaioglu, A. S., Dalkıran, A., Cetin Atalay, R., Zhang, C., Hurto, R. L., Freddolino, P. L., … Friedberg, I. (2019). The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. Genome Biology, 20(1), 244. https://doi.org/10.1186/s13059-019-1835-8

[48] Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J. P., & Tamayo, P. (2015). The Molecular Signatures Database Hallmark Gene Set Collection. Cell Systems, 1(6), 417–425. https://doi.org/10.1016/j.cels.2015.12.004

[49] Kinsella, R. J., Kähäri, A., Haider, S., Zamora, J., Proctor, G., Spudich, G., Almeida-King, J., Staines, D., Derwent, P., Kerhornou, A., Kersey, P., & Flicek, P. (2011). Ensembl BioMarts: A hub for data retrieval across taxonomic space. Database, 2011, bar030. https://doi.org/10.1093/database/bar030

[50] Joachimiak, M. P., Caufield, J. H., Harris, N. L., Kim, H., & Mungall, C. J. (2024). Gene set summarization using large language models. arXiv. https://doi.org/10.48550/arXiv.2305.13338

[51] [39] Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. arXiv. https://doi.org/10.48550/arXiv.2102.07350

[52] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9), 1–35. https://doi.org/10.1145/3560815

[53] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. arXiv. https://doi.org/10.48550/arXiv.2201.11903

[54] Zheng, Y., Zhang, R., Zhang, J., Ye, Y., Luo, Z., Feng, Z., & Ma, Y. (2024). Llama-Factory: Unified efficient fine-tuning of 100+ language models. arXiv. https://doi.org/10.48550/arXiv.2403.13372

[55] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (Vol. 2, pp. 1137–1143). https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf

[56] Prechelt, L. (1998). Early stopping—But when? In G. B. Orr & K.-R. Müller (Eds.), Neural networks: Tricks of the trade (pp. 53–67). Springer. https://doi.org/10.1007/3-540-49430-8_3

[57] Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In S. Dasgupta & D. McAllester (Eds.), Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28, pp. 1310–1318). PMLR. https://proceedings.mlr.press/v28/pascanu13.html

[58] Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. arXiv. https://doi.org/10.48550/arXiv.1608.03983

[59] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. arXiv. https://doi.org/10.48550/arXiv.1711.05101

[60] Kalajdzievski, D. (2024). Scaling laws for forgetting when fine-tuning large language models. arXiv. https://doi.org/10.48550/arXiv.2401.05605

[61] Ren, W., Li, X., Wang, L., Zhao, T., & Qin, W. (2024). Analyzing and reducing catastrophic forgetting in parameter efficient tuning. arXiv. https://doi.org/10.48550/arXiv.2402.18865

[62] Liu, C., Kang, Y., Wang, S., Qing, L., Zhao, F., Sun, C., Kuang, K., & Wu, F. (2024). More than catastrophic forgetting: Integrating general capabilities for domain-specific LLMs. arXiv. https://doi.org/10.48550/arXiv.2405.17830

[63] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611–626). Association for Computing Machinery. https://doi.org/10.1145/3600006.3613165

[64] React. (n.d.). React (Version 19.2) [Computer software]. Retrieved March 12, 2026, from https://react.dev/

[65] Unstructured-IO. (n.d.). Unstructured (Version 0.21.5) [Computer software]. GitHub. https://github.com/Unstructured-IO/unstructured

[66] National Center for Biotechnology Information. (n.d.). Entrez Programming Utilities (E-utilities) [API]. U.S. National Library of Medicine. https://www.ncbi.nlm.nih.gov/home/develop/api/

[67] Hendricks, G., Tkaczyk, D., Lin, J., & Feeney, P. (2020). Crossref: The sustainable source of community-owned scholarly metadata. Quantitative Science Studies, 1(1), 414–427. https://doi.org/10.1162/qss_a_00022

[68] Europe PMC Consortium. (2015). Europe PMC: A full-text literature database for the life sciences and platform for innovation. Nucleic Acids Research, 43(D1), D1042–D1048. https://doi.org/10.1093/nar/gku1061

[69] Unpaywall. (n.d.). REST API [Computer software]. Retrieved March 12, 2026, from https://unpaywall.org/products/api

[70] PubMed Central. (n.d.). PMC Open Access Web Service API [Computer software]. Retrieved March 12, 2026, from https://pmc.ncbi.nlm.nih.gov/tools/oa-service/

[71] Hugging Face. (n.d.). TRL - Transformer Reinforcement Learning [Documentation]. Retrieved March 12, 2026, from https://huggingface.co/docs/trl/index

[72] OpenAI. (n.d.). GPT-5 mini [Large language model]. OpenAI API. Retrieved March 12, 2026, from https://developers.openai.com/api/docs/models/gpt-5-mini

[73] OpenAI. (2025, August 13). GPT-5 system card. https://cdn.openai.com/gpt-5-system-card.pdf

[74] Zhu, X., Chen, T., Yang, H., & Lv, K. (2020). Lactate induced up-regulation of KLHDC8A (Kelch domain-containing 8A) contributes to the proliferation, migration and apoptosis of human glioma cells. Journal of Cellular and Molecular Medicine, 24(20), 11691–11702. https://doi.org/10.1111/jcmm.15780

[75] Lee, D., Gimple, R. C., Wu, X., Prager, B. C., Qiu, Z., Wu, Q., Daggubati, V., Mariappan, A., Gopalakrishnan, J., Sarkisian, M. R., Raleigh, D. R., & Rich, J. N. (2023). Superenhancer activation of KLHDC8A drives glioma ciliation and hedgehog signaling. The Journal of Clinical Investigation, 133(2), e163592. https://doi.org/10.1172/JCI163592

[76] Farlie, P. G., Reid, C., Wilcox, S., Peeters, J., Reed, G., & Newgreen, D. F. (2001). Ypel1: A novel nuclear protein that induces an epithelial-like morphology in fibroblasts and is expressed in the developing face. Genes to Cells, 6(7), 619–629. https://doi.org/10.1046/j.1365-2443.2001.00445.x

[77] Abiatari, I., Kiladze, M., Kerkadze, V., Friess, H., & Kleeff, J. (2009). Expression of YPEL1 in pancreatic cancer cell lines and tissues. Georgian Medical News, (175), 60–62. https://pubmed.ncbi.nlm.nih.gov/19893129/