University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

AI-based Generation of Descriptions for Protein Signatures: Fine-Tuning of Large Language Models and Comparative Analysis

Supervisor
Prof. Damiano Piovesan
University of Padova

Master Candidate
Angela Kralevska

Academic Year
2024-2025

"Artificial intelligence is not a substitute for human intelligence; it is a tool to amplify human creativity and ingenuity."
— Fei-Fei Li

Abstract

One of the most important aspects of biological research is functional classification and annotation of protein sequences. InterPro is a database that categorizes protein sequence patterns, known as "signatures", and provides models for automatically classifying new protein sequences. Moreover, most of the InterPro entries are associated with detailed functional information, which is standardized by employing Gene Ontology (GO) terms.

While GO terms have proven to be very useful for machine-related tasks, for researchers that work with biological data, it is very important to have concise, human-readable, and understandable descriptions for the biological entities (proteins, families, domains, sites) under study. The purpose of this project is to automate the generation of short functional descriptions for different types of biological entities by using natural language generation models. In particular, we focused on those signatures corresponding to intrinsically disordered regions, i.e. protein fragments that are missing a fixed three-dimensional structure and have poor functional characterization.

To achieve our goal, different datasets with varying levels of complexity were constructed from InterPro entries. Simpler datasets contain GO terms as input, while more complex datasets combine GO terms and functional descriptions extracted from the literature, as reported in the UniProtKB database, of the proteins associated with a given InterPro signature, as well as the corresponding protein organism.

We fine-tuned the T5, GPT-2, and BioGPT large language models. Notably, BioGPT is a model that is pre-trained using large-scale biomedical literature. Our findings demonstrate that the fine-tuned T5 model significantly outperforms the pre-trained T5 model, and it is also superior compared to the fine-tuned GPT-2 and BioGPT models. Best results were achieved when the model was fine-tuned with the most complex dataset type.

The best model was further tested on a novel task: generating descriptions for intrinsically disordered regions, which were not part of the training data, highlighting its adaptability but also revealing areas for improvement.

Contents

Listing of figures

Listing of tables

Listing of acronyms

GO ..... Gene Ontology
DAG ..... Directed Acyclic Graph
LLM ..... Large Language Model
PTM ..... Post-Translational Modification
IDP ..... Intrinsically Disordered Protein
IDR ..... Intrinsically Disordered Region
IDPO ..... Intrinsically Disordered Proteins Ontology
ECO ..... Evidence and Conclusion Ontology
AI ..... Artificial Intelligence
NLP ..... Natural Language Processing
NLG ..... Natural Language Generation
NER ..... Named Entity Recognition
QA ..... Question Answering
RNN ..... Recurrent Neural Network
T5 ..... Text-to-Text Transfer Transformer
GPT ..... Generative Pre-Training Transformer
C4 ..... Colossal Clean Crawled Corpus
LM ..... Language Modeling
CLM ..... Causal Language Modeling
MLM ..... Masked Language Modeling
DAE ..... Denoising Autoencoder
WMD ..... Word Mover's Distance
WMS ..... Word Mover's Similarity
SMS ..... Sentence Mover's Similarity
W+SMS ..... Sentence and Word Mover's Similarity
BOW ..... Bag-Of-Words
STS ..... Semantic Textual Similarity
SBERT ..... Sentence-BERT
NLI ..... Natural Language Inference
AS ..... Abstractive Summarization
BLAST ..... Basic Local Assignment Search Tool

1 Introduction

Proteins are one of the most important macromolecules in living organisms that drive nearly every biological process, including reaction catalysis, cell shape maintenance, cell interactions, and many others [5][6]. Understanding the functional roles of the proteins in those processes is very important. They can be classified into protein families, domains, and other groups based on their sequence and structural characteristics by using diverse methods known as "signatures". Protein classification is an important aspect of modern genomics since it allows researchers to infer functional properties of novel proteins by associating them with well-characterized groups. InterPro is a database that combines data from multiple protein signature databases into a single resource. Many of its entries, which can be protein domains, families, active or binding sites, post-translational modification sites, repeats, etc., contain short descriptions that describe their functions and characteristics. The majority of them are curated and manually written by experts, but there exist some that are AI-generated and properly marked so they can be distinguished from the curated ones [1]. Those descriptions allow researchers to capture various aspects of protein functions.

However, these descriptions are not suitable for computational analysis because computers require information to be encoded and standardized. That is why the Gene Ontology (GO), Intrinsically Disordered Proteins Ontology (IDPO), and other types of ontologies have been created. They represent a structured vocabulary that standardizes the description of protein functions [7]. GO terms have enabled protein function data to be used for many computational tasks that commonly involve artificial intelligence methods, such as function prediction [8]. While GO has made possible significant progress in many bioinformatics tasks, the human-readable short descriptions are much more convenient for interpretation by the researchers. Having concise and informative descriptions that summarize the functional attributes of proteins is very important for the biologists. Nevertheless, the manual effort required to write these descriptions can be time-consuming for the biological curators, given the fact that massive amounts of new data are being produced by computational approaches nowadays.

The aim of this thesis is to automate the generation of these functional descriptions, transforming GO terms and related protein information into readable summaries for the protein signatures. There have been huge ad-

vancements in the natural language processing (NLP) field in the last few years. Many large language models (LLMs) with outstanding capabilities have been developed, making it feasible to generate coherent and meaningful descriptions [9]. These models have the potential to speed up the work of biological curators, which would imply improved accessibility to protein function information for researchers. The work in my thesis focuses on fine-tuning pre-trained language models to make them capable of generating descriptions for protein domains, families, and sequence features, including the intrinsically disordered regions (IDRs). IDRs are parts of the protein sequences that do not fold into stable 3D structures. They play an essential role in the regulation of signaling pathways and many cellular processes. Despite their importance, just a small number of IDRs have been functionally studied, and the leading resource for collecting information about them is the DisProt database [10][11].

By fine-tuning three types of pre-trained models with datasets of diverse complexity that combine GO terms and UniProtKB protein annotations, we aim to speed up the process of generating functional descriptions by providing AI-generated summary proposals. As a result, we intend to bridge the gap between machine-readable representations and human-readable summaries, which would be of great help to the scientific community.

1.1 Proteins

Proteins are one of the most important macromolecules found in all living organisms. They are polymers composed of amino acids linked by peptide bonds. All proteins are made from a selection of twenty amino acids, which are joined into long polypeptide chains. The amino acid sequence of a protein provides information about its three-dimensional structure and function. Proteins can be small or large, hydrophilic or hydrophobic, isolated or coupled with other molecules such as ligands or carbohydrates, and so on. Their key differentiating feature is the precise selection of amino acids, as well as the order in which they follow one another in the polypeptide chain, which is genetically determined [5].

The precise sequence of amino acids allows proteins to fold up into a particular three-dimensional shape, called conformation. Fully folded proteins have distinct surface properties that influence which molecules they can interact with. When proteins interact with other molecules, their shape may change in subtle or significant ways [12]. Protein binding is very important for the function of the proteins. They are involved in almost all interactions and reactions that take place in living organisms [6].

1.1.1 Protein Structure

Proteins fold into three-dimensional structures that are determined by their amino acid sequence. The folding of the protein is essential for its function. In general, there are four levels of protein structure: primary, secondary, tertiary, and quaternary. They are shown in Figure 1.1.

The primary structure is the amino acid sequence of the polypeptide chain. It determines how the protein folds into a unique three-dimensional structure. The primary structure is held together primarily by peptide bonds, as well as disulfide bonds.

The secondary structure of a protein is defined as the local spatial arrangement of its amino acid residues within a polypeptide chain. The two most common types of secondary structure found in proteins are α-helices and β-sheets. In a α-helix, the polypeptide chain coils into a right-handed spiral, with hydrogen bonds between every

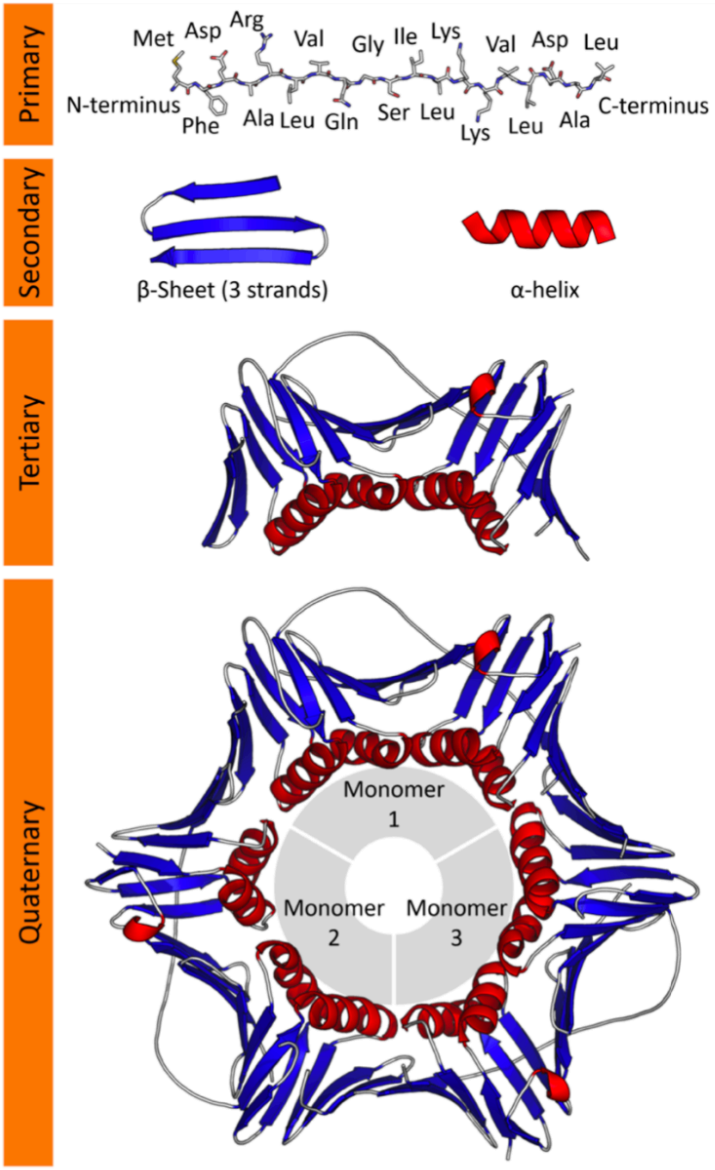

Figure 1.1: Overview of the four levels of protein structure (primary, secondary, tertiary, quaternary) using the 1AXC protein as an example. License: Creative Commons Attribution 4.0 International. Created by Thomas Shafee.

fourth amino acid. This regular arrangement ensures stability and rigidity. In contrast, a β-sheet is made up of β-strands, which are extended polypeptide segments that align side by side. Hydrogen bonds form between the strands, resulting in a sheet-like structure. A β-sheet's strands can be parallel or antiparallel, based on the directionality of the chains.

Tertiary structure describes the three-dimensional folding of a polypeptide. The protein's folded shape allows it to interact with other molecules in a very specific way. This is necessary for the protein to perform its biological functions effectively. There are two types of proteins based on tertiary structure: fibrous and globular. Fibrous proteins, which primarily serve structural roles, contain simple repeating secondary structure elements. Globular proteins have more complex tertiary structures, which frequently include multiple types of secondary structure in the same polypeptide chain.

The quaternary structure is a large assembly of multiple protein molecules or polypeptide chains, which are commonly referred to as subunits. Multimers are complexes that contain multiple subunits. Some proteins do not have a quaternary structure and instead function as monomers. Understanding the quaternary structure is important for understanding how specific proteins are involved in cellular processes and interactions [13].

It is widely assumed that a protein's function is closely linked to its three-dimensional structure, and that is why prediction or experimental determination of the protein structures is a matter of high priority. However, it has been discovered that a large number of gene sequences code for long stretches of amino acids that are likely to unfold in solution or adopt non-globular structures of unknown conformation [14]. The proportion of these types of sequences in all studied organisms' genomes is very high, and even though they do not adopt stable, globular conformations, they still have important functional roles. They are known as intrinsically disordered proteins and regions, and we will discuss them in the following sections.

1.1.2 Protein Function

Proteins are extremely diverse and perform a variety of functions. They have a wide range of roles in the cells, including maintaining cell shape, structural organization, acting as catalysts, molecular sensors, aiding in cell interactions, movement of the cell, and providing a cell defense mechanism, among others [5].

Based on the DNA content of the human body, it has been estimated that it contains approximately 100.000 distinct protein types. Proteins are typically unique to the species from which they are derived. While they may be structurally and functionally similar, their compositions vary slightly, leading to them being immunologically distinct. This demonstrates that proteins must exist in an almost infinite number of variations and perform a broad range of tasks [15].

Proteins can be enzymes, which are catalysts that help chemical reactions occur faster in the body and thus allow some vital processes like metabolism or digestion to happen more quickly. Some of them are responsible for moving chemicals within and between cells, enabling processes such as nutrient intake and waste elimination. Proteins also play an important role in cell signaling and communication. They act as receptors on cell surfaces, allowing them to respond to external signals like neurotransmitters and hormones. Some of them also help maintain the structural integrity of cells, tissues, and organs. Regulatory proteins control gene expression, ensuring that the correct genes are active at the appropriate times. Motor proteins control cell mobility and organelle transport within cells. Antibodies are proteins that help the immune system recognize and eliminate external invaders such as bacteria and viruses [12].

Proteins can be generally classified into four types: globular, membrane, fibrous, and intrinsically disordered proteins. Protein characteristics differ based on the protein type. Membrane proteins typically have higher hydrophobicity compared to other protein types. This nature is strongly related to their roles in the cell. Proteins from various types can be fused. For example, a protein can have both soluble and transmembrane domains, as well as soluble and intrinsically disordered regions. The features of individual domains determine the overall protein function [16].

1.1.3 Gene Ontology

The previous paragraphs described how characterizing protein functions is one of the most important problems in biology. This paragraph discusses a method of representing the function.

Functional descriptions of proteins can be written in textual form, allowing biologists to elaborate on numerous aspects of protein functions. The textual descriptions are available in many public databases [17]. However, this form of representation has the downside that it cannot be used easily by the computer systems since it is not structured and standardized. To solve this problem, researchers have created an ontology, which is a structured method for representing knowledge within a particular domain.

The Gene Ontology (GO) resource, established in 1998, provides structured knowledge about the functions of genes and gene products [7]. It is a standardized representation framework used in bioinformatics and genomics to categorize and describe the functions of genes and gene products across different organisms. Its contents are constantly improving, both in quantity and quality [18].

The ontologies are commonly represented as graphs, with nodes indicating terms or concepts in the field of interest and edges representing the relationships between them. While there are no strict guidelines on how ontologies must be implemented, many are represented as directed acyclic graphs (DAGs) [19]. In GO, the classes of gene functions, known as GO terms, are systematically related to one another. As of the 2024 release, there are approximately 40 000 terms in GO, linked by more than 100 000 relationships [*].

The three primary categories of Gene Ontology terms are:

- **Biological Process** - defines a variety of broader biological events or processes mediated by one or more gene products. It describes the roles of genes in a number of cellular activities, including metabolism, signal transduction, cell division, etc.
- **Molecular Function** - corresponds to specific biochemical activities that occur at the molecular level, such as enzyme catalysis, transcription factor activity, and receptor binding.
- **Cellular Component** - refers to the cellular context in which the gene's function is carried out. It focuses on the subcellular structures or sites where a gene product is located, such as the nucleus, mitochondria, or another organelle.

Each GO term may be related to its parent through a different relationship type. Some examples include: "is-a" relationships, which indicate that one term is a subtype or a more specific instance of another term; "part-of" relationships, which define that the term is a structural or functional part of the term on the other side; and "regulates" relationships, which describe how one term regulates the activity of another term, etc. One example of biological process and the relationships with its ancestors are represented on Figure 1.2 [†].

[*] Retrieved from: https://geneontology.org/stats

[†] Retrieved from: https://www.ebi.ac.uk/QuickGO

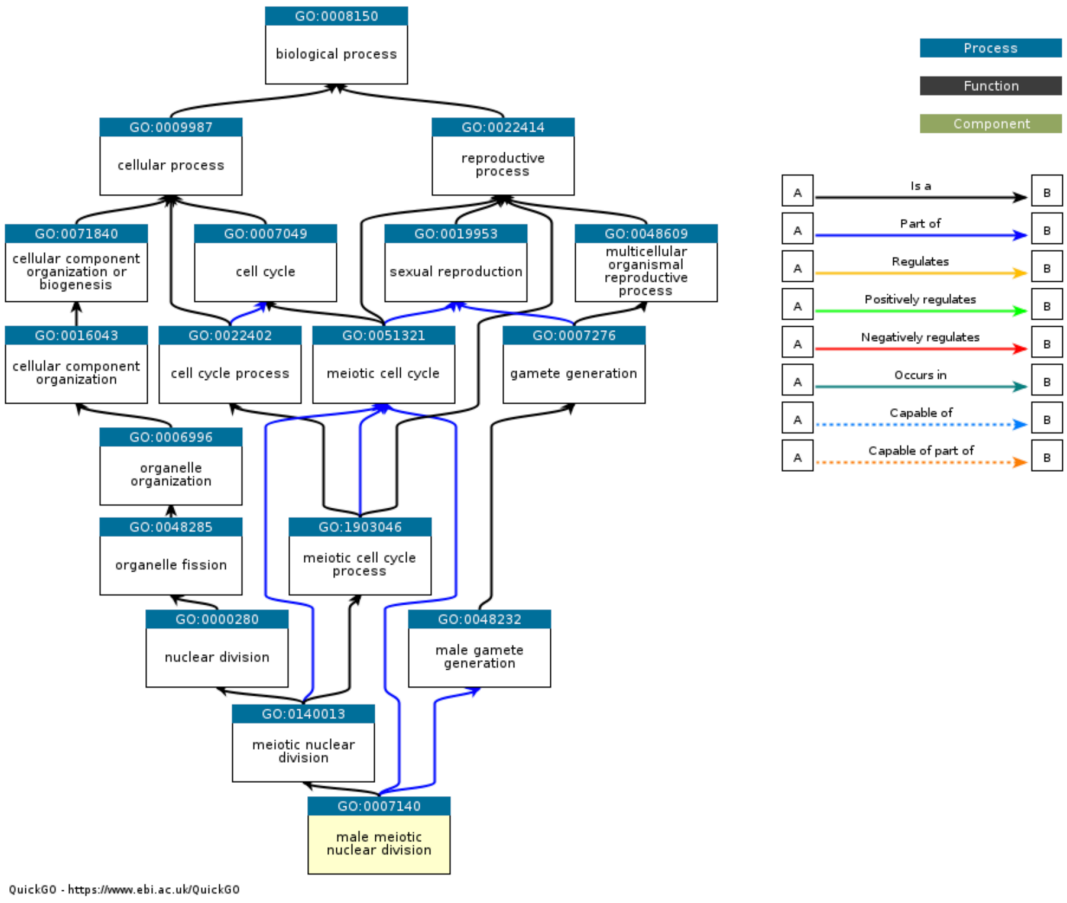

Figure 1.2: A visual representation of the male meiotic nuclear division biological process and its connections to all of its predecessors.

The GO knowledgebase includes annotations that relate individual gene products from organisms across the tree of life to ontology terms. Each annotation is labeled with the underlying evidence, such as a peer-reviewed paper. The GO knowledgebase has about 7 million annotations to genes/gene products from over 3,200 species, with around 10% supported by experimental data from research papers [18]. GO has significantly smoothed the path of computational studies of protein functions, such as protein function prediction, GO enrichment analysis, and others, which are essential for genomics and proteomics studies [20][8].

1.1.4 UniProt Knowledgebase

The UniProt Knowledgebase (UniProtKB) aims to provide users with a comprehensive, high-quality, and freely available set of protein sequences annotated with functional information. In 2022, UniProtKB contained more than 227 million sequences. Detailed annotations from the literature are continuously extracted in order to up-

date or produce reviewed entries, while the unreviewed entries are supplemented with annotations generated by automated systems employing various kinds of machine-learning methods.

UniProtKB consists of two parts: the reviewed protein set (UniProtKB/Swiss-Prot), in which each protein entry is linked to a summary of the experimentally verified or computationally predicted functional information added by their biocuration team, and the unreviewed protein set (UniProtKB/TrEMBL), in which entries are automatically annotated by computer systems.

The summarizing of biological data obtained from the scientific literature remains essential for UniProtKB. The knowledgebase provides both human-readable free-text summaries and structured information that can be used for large-scale analysis as well as the creation of training data sets for AI-based method development. As machine learning methods improve, the team at UniProtKB is exploring how to integrate these approaches in the curation process and also use them to automate the annotation of proteins that have not been experimentally studied [21].

1.2 Protein Classification

Proteins can be classified into groups based on their sequence or structural similarities. These groups often include well-characterized proteins with known functions. When a novel protein is discovered, its functional properties can be inferred by examining the group to which it is predicted to belong. Protein classification into groups can be based on several aspects, including the families they belong to, the domains they contain, and their sequence features. These groups, referred to as biological entities or sequence signatures, include domains, families, sequence features, sites, homologous superfamilies, and intrinsically disordered regions [1]. They will be described in more detail in the following paragraphs.

1.2.1 Protein Domains

There is no universally accepted definition of a protein domain, mainly because of the subjective nature of domain assignment and categorization. When they are described at the sequence level, protein domains are homologous portions of sequences that, despite appearing in different gene contexts, have remained conserved throughout evolution. Domains are considered the ”evolutionary units” of protein sequences. There is evidence that a large number of genes consist of multiple protein domains [22].

When it comes to structure, domains are commonly defined as local, compact structural units with a hydrophobic interior and a hydrophilic exterior that form a globular-like state that cannot be further subdivided. Also, they are commonly referred to as semi-independent globular folding units. This characteristic has allowed them to effectively integrate with other domains and evolve new functions. Domain assignment offers an approximate understanding of protein function, implying that the overall function is a combination of the individual parts. This view of protein function is sometimes overly simplistic, particularly when experimental data is considered, but it still provides useful insights [23]. Domains can exist in a variety of biological contexts, and similar domains can be found in proteins with different functions [24].

1.2.2 Protein Families

A protein family is a group of proteins that have a common evolutionary origin, which is evident in their similar functions and structural or sequence characteristics. These families can be defined as groups of molecules with significant sequence similarities among their members [25]. Particularly, proteins from the same family frequently have similar or sometimes even identical biochemical functions, highlighting their common origin [26].

Throughout evolution, proteins derived from a common ancestral protein were able to diverge due to mutations in their sequences, resulting in families of homologous proteins. Not all positions in the sequences are equally susceptible to mutation, since some of them can be important for the protein function, stability, or folding. Many protein family resources use a hierarchical classification system in which close relatives with high sequence similarity, i.e., more than 40% identity of the sequence, are placed together into families. These close relatives usually have common functions. Remote homologues with low sequence similarity, usually less than 30% sequence identity, are grouped into larger evolutionary families or superfamilies. When classifying proteins into hierarchical families, the level at which we can place a protein in the hierarchy is very important because it determines the amount of specific functional information that can be inferred [27][28].

Protein family classification is a key technique for large-scale genomic annotation. It helps the identification of proteins that are difficult to characterize through pairwise alignments. Proteins that are thoroughly studied within a family can help reliably assign functions to other family members whose roles are either unknown or unclear [29]. Additionally, it helps organize databases by enabling family-based annotation propagation and identifying potential annotation errors. This classification also aids in extracting relevant biological information from large datasets. Finally, it reflects underlying gene families, which are very important for phylogenetics and comparative genomics [30].

Figure 1.3 shows the domain organization of proteins with RGS domains across three families: RGS proteins, beta-adrenergic receptor kinases, and sorting nexins. Each family has distinct domain combinations, some proteins contain only the RGS domain, while others have multiple domains.

1.2.3 Protein Sequence Features

Sequence features are short sequences of amino acids that play an important role in determining specific protein characteristics, which are often meaningful for the overall function of the protein. Unlike domains, which are larger structural or functional units of a protein, sequence features are typically much smaller, sometimes only a few amino acids long, and can be found within domains. Proteins can also be classified based on the sequence features they contain [1]. Some types of sequence features are shown on Figure 1.4 and will be described in more detail in the following paragraphs.

Active Sites

The active site is the region of an enzyme where substrate molecules interact and undergo a chemical reaction. The active site is the most significant region of an enzyme since it directly catalyzes the chemical reaction, although it takes up only around 10–20% of the enzyme's volume. It is typically composed of three to four amino acids, while the other amino acids are required to maintain the tertiary structure of the enzymes [31][32].

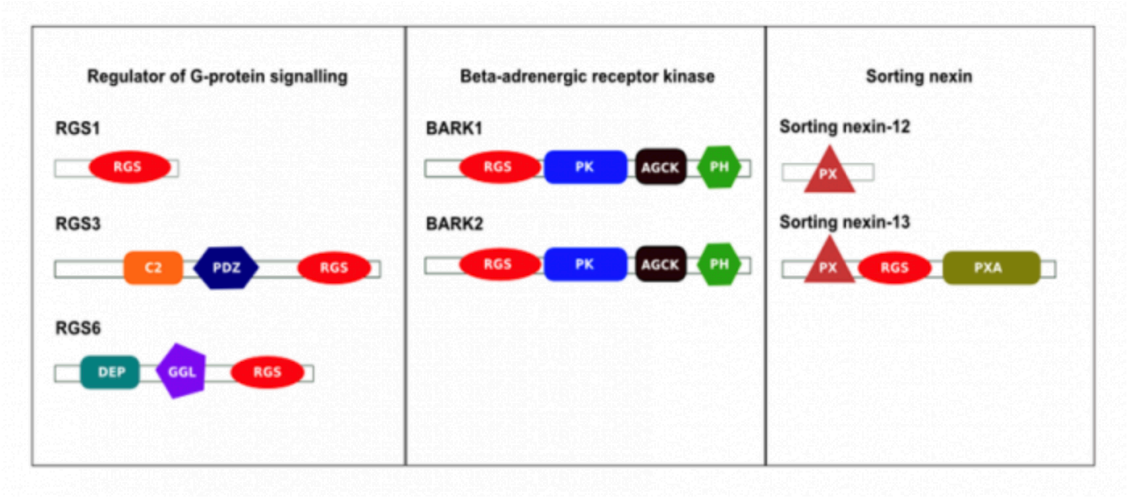

Figure 1.3: Domain organization in proteins containing RGS domains across different families. License: Creative Commons Attribution 4.0 International. Reproduced from InterPro [1].

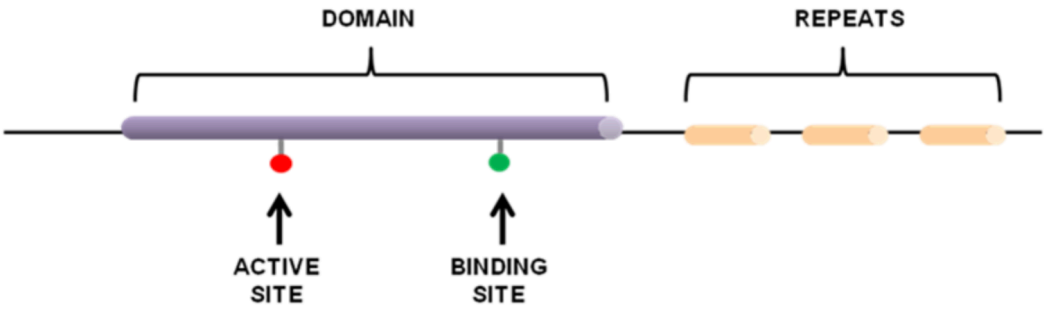

Figure 1.4: Illustration of the repeats, domains, and sites within a protein sequence. License: Creative Commons Attribution 4.0 International. Reproduced from InterPro [1].

Binding Sites

A binding site is a region in a macromolecule, such as a protein, that binds to other molecules or ions [33].

Post-Translational Modification Sites

Protein post-translational modifications (PTMs) represent the breaking or formation of covalent bonds on the backbones or amino acid side chains of proteins, which increases protein variety and sets the basis for the emergence of organismal complexity. Over 650 types of protein modifications have been identified, including phosphorylation, ubiquitination, methylation, etc. The list continues to grow [34].

Repeats

Repeats are short amino acid sequences that occur several times within a protein and may confer binding or structural characteristics. An array of protein tandem repeats consists of multiple adjacent copies with the same or similar sequence motifs. These periodic patterns are caused by internal duplications in both coding and non-coding genomic sequences. Protein tandem repeats can be diverse, starting from a single amino acid to domains containing larger number of residues [35][36].

1.2.4 InterPro Database

InterPro is a database that categorizes proteins into families and predicts domains and sites using predictive models from numerous member databases that are combined into a single resource [1].

Rapid developments in genomic technologies, combined with considerable reductions in sequencing costs, have enabled researchers to generate sequencing data at an unprecedented rate [37]. Large-scale sequencing data must be analyzed and characterized before it can be used by the scientific community. However, this presents a challenge because the processing time required for analysis is growing exponentially. To overcome this challenge, various automated methods have been developed to analyze sequences and annotate protein families, domains, and functional sites. These methods often transfer information from experimentally characterized sequences to uncharacterized ones using predictive models like hidden Markov models, patterns, profiles, or fingerprints, which are collectively referred to as signatures. Several protein signature databases have been established, each focusing on specific areas, and InterPro combines 13 of those to a single resource [1]. It also provides annotations from other tools and resources besides the member databases. These resources include MobiDB-lite for disordered regions [38].

When multiple member database signatures refer to the same biological entity, they are integrated into a single InterPro entry to reduce redundancy. Each InterPro entry is assigned a unique name, short name, and accession number, as well as a descriptive abstract and GO terms. Before being made public, new InterPro entries undergo curators review. Some entries of the type ”family” have AI-generated names, short-names and descriptions by using a Large Language Model (LLM). The content that is generated by LLMs is flagged with specific tags in InterPro so the users can distinguish it from the human generated content [1].

1.3 Intrinsically Disordered Proteins and Regions

Intrinsically disordered proteins (IDPs) have a biased amino acid composition with low sequence complexity. They contain fewer bulky hydrophobic amino acids and higher amounts of charged or hydrophilic ones. Although these proteins are functional, they do not fold into stable, fixed tertiary structures like other proteins. Instead, they remain flexible and continuously change shape, ranging from extended coil-like forms to collapsed globules [39].

Some proteins are anticipated to be completely disordered, whereas others have disordered parts of their sequences known as intrinsically disordered regions (IDRs) and structured globular domains. The largest number of proteins in eukaryotic proteomes comprise both structured regions and IDRs [40]. The majority of protein disorder occurs in the IDRs which function as linkers between two folded domains, and in the ones that act as tails located at the protein's termini. Approximately 70% of proteins in the human proteome contain one or more IDRs of 30 or more residues [41]. On Figure 1.5 is given an illustration of IDP, protein with IDRs, and structured protein.

IDPs commonly interact with or serve as hubs in protein interaction networks [42]. Despite the lack of a fixed 3D structure, IDRs are crucial for cellular function. They play an important role in the regulation of signaling pathways and essential cellular processes, such as transcription, translation, cell cycle, etc [43][44]. The levels of IDPs in cells are carefully maintained to ensure appropriate signaling in both time and space, and the mutations or changes in their abundance frequently are associated with diseases [45]. On the molecular level, IDRs can function as flexible linkers, adaptable modules for molecular recognition, binding sites for many partners, cellular sensors, and subcellular structure organizers[40]. Their lack of a fixed structure, combined with their ability to adopt various conformations, allows them to interact with other IDRs, structured proteins, and nucleic acids through diverse multivalent interactions [46]. Despite their crucial roles, just a small number of IDRs have been functionally studied, with most information hidden deep in the scientific literature [10][11].

1.3.1 Intrinsically Disordered Proteins Ontology

The DisProt consortium maintains the Intrinsically Disordered Protein Ontology (IDPO). Although GO is intended to annotate the entire protein, only some of its terms, particularly those in the molecular function branch, are well suited to characterize IDR behavior. Specific IDPO terms are used to annotate self-regulatory functions as well as the functions that arise directly from the disordered state because equivalent terms are not yet available in GO.

The IDPO comprises three major branches, each reflecting a different component. The three branches, referred to as "namespaces," are structural state, structural transition, and disorder function. The structural state represents a state occupied by a particular protein region, such as disorder, pre-molten globule, molten globule, and order. The structural transition namespace contains child terms that reflect possible transitions between different structural states, such as disorder-to-order and order-to-disorder. The disorder function namespace consists of both self-functions and functions related to the ability of disordered proteins and regions to switch between conformational states [47].

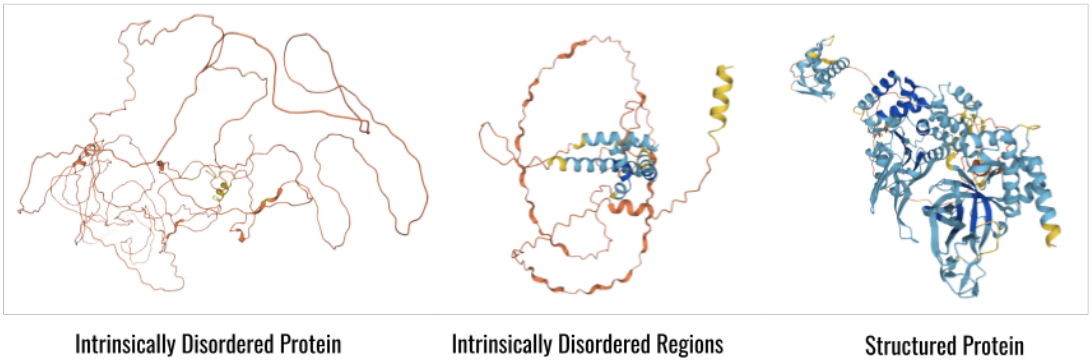

Figure 1.5: Visual representation of an IDP, a protein with IDRs, and a structured protein. From left to right, UniProtKB IDs: P04985, P08453, and Q9XYU1. Structures generated using AlphaFold predictions [2].

1.3.2 DisProt Database

The previous paragraphs explained the importance of IDRs and pointed out that their characterization is very challenging. The main issue is that disorder cannot be described as a simple binary state but rather as an ensemble of heterogeneous structures with different properties and functions that often depend on the context. These properties only appear under certain conditions like pH, localization, binding, or post-translational modifications [48]. To help the researchers, it is very important to collect standardized and accessible experimental data on IDPs, and the DisProt database has been doing this since it was first released in 2005 [49].

DisProt, or the Database of Intrinsically Disordered Proteins, is the leading repository for manually curated annotations of intrinsically disordered proteins and regions acquired from scientific literature. It is a reliable resource for understanding the functions of disordered proteins, providing a variety of experimentally validated data. By consolidating evidence on protein disorder, it can significantly help researchers to enhance their knowledge of these proteins and their funcions in various biological processes [47].

DisProt uses three ontologies to annotate intrinsically disordered regions: the Intrinsically Disordered Proteins Ontology (IDPO), Gene Ontology (GO), and Evidence and Conclusion Ontology (ECO). As we previously stated, IDPO is used to characterize structural aspects of the intrinsically disordered proteins and regions, including self-functions and functions that are directly related to their disordered state. GO is used to characterize the functional components, while ECO refers to the technique or evidence connected with an annotation [47].

1.4 Natural Language Processing Models

Machine and deep learning are powerful methods for unraveling the complexities of biological data, producing promising results in a variety of bioinformatics tasks. In recent years, there have been huge advancements in the Natural Language Processing (NLP) field. The development of the transformer models [3] revolutionized NLP by introducing the "attention mechanism," which allows the models to weigh the relevance of each word in a sentence relative to the other words rather than processing them in a predetermined order. NLP models are used

to analyze and interpret large volumes of biological information, allowing researchers to gain useful insights from literature, annotations, and other text sources relevant to genomics and proteomics. Aside from this, there are models that can generate text on their own, so-called Natural Language Generation (NLG) models.

At present, the most recent LLMs that have been released, such as GPT-4 [50], Claude [51], and Gemini [52], have trillions of parameters and are exceptionally good for various use cases in many diverse fields. Typically, they are used by constructing and sending them specific prompts for the problem that we want to address, a process known as prompt engineering. Additional fine-tuning for them in most cases is not required. Unfortunately, the best-performing LLMs are not open source, making them unsuitable for academic research. This is one of the reasons why fine-tuning smaller pre-trained models with fewer parameters, such as BERT [53], GPT-2 [54], T5 [55], and so on, may still be preferable in some cases. Pre-trained LLMs are typically trained on general domain corpora, namely texts from numerous sources available on the internet. In many cases, fine-tuned models pre-trained on domain-specific data, such as biological data, tend to perform better in downstream biological tasks than the fine-tuned models pre-trained with general data. Several models have been pre-trained using biological data. Some of these include BioBERT [56], PubMedBERT [57], BioGPT [4], and ProtLLM [58].

BioBERT is a model that was pre-trained on large-scale biomedical data. It has outperformed BERT and previous state-of-the-art models in a wide range of biomedical text mining tasks, including biomedical named entity recognition (NER), biomedical relation extraction, and biomedical question answering (QA) [56]. The creators of the PubMedBERT model have challenged a prevailing assumption in pretraining neural language models and demonstrated that domain-specific pre-training from scratch in the biomedicine field can significantly outperform continual pre-training from a general-domain language model, resulting in new state-of-the-art results for a wide range of biomedical NLP applications [57]. While BioBERT and PubMedBERT have had considerable success in a range of discriminative downstream biomedical tasks, their absence of capability to generate text limits their application area. That is why BioGPT, a domain-specific generative Transformer language model, has been proposed. The authors took GPT-2 as a backbone model and performed training on a corpus of 15 million PubMed abstracts. They have evaluated BioGPT on six biomedical natural language processing tasks and demonstrated that their model outperformed state-of-the-art models on end-to-end relation extraction, question answering, document classification, and text generation tasks. It has also shown better performance than GPT-2 in the biomedical text generation task [4].

In the following paragraphs, the transformer architecture will be described in more detail, followed by some specifics for the models that we were using in our study.

1.4.1 Transformers Architecture

A team of researchers has introduced the transformer architecture in a 2017 study titled "Attention Is All You Need" [3]. The introduction of transformers was an important turning point in NLP, paving the way for LLMs, which have since demonstrated remarkable progress, outperforming previous models that were considered cutting-edge, such as recurrent neural networks (RNNs).

The development of the transformer architecture was inspired by RNN's encoder-decoder architecture. However, instead of relying on recurrence, transformers use an attention-based mechanism exclusively. The transformer architecture changed the way neural networks handle sequences by introducing the self-attention concept. This self-attention mechanism allows each word in the sequence to be related to the complete context of the

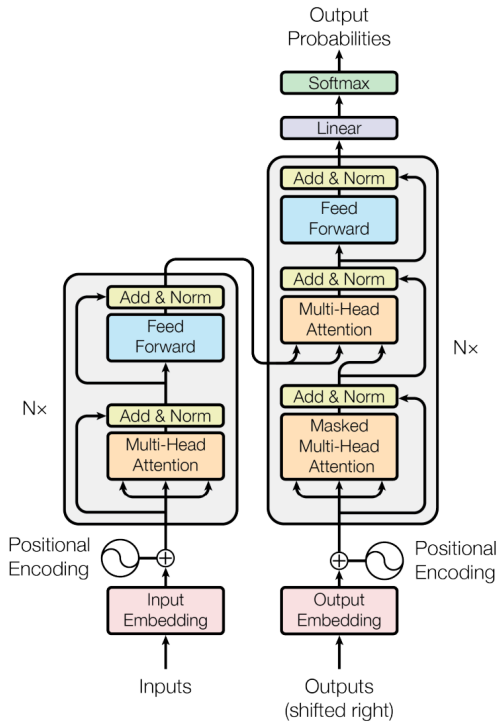

Figure 1.6: Transformer architecture proposed in [3].

sentence rather than just the preceding words. The transformer architecture has been explained in detail in the original paper [3], and we will just provide a brief summary of some of its components here.

Encoder/Decoder Architecture

The transformer architecture includes two main components: the encoder and decoder. They are represented in Figure 1.6. The encoder is a crucial component of the transformer architecture. Its function is to convert input tokens into contextualized representations. Unlike previous models, which processed tokens independently, the transformer's encoder preserves the context of each token with regard to the entire sequence. The encoder generates representations, which are then decoded to produce the output text. The encoder and decoder are trained simultaneously in order to maximize the conditional log-likelihood of the output given the input.

In the original transformer architecture, the encoder and decoder had six identical layers. All encoder layers contained two sub-layers, which are a multi-head self-attention layer and a feed-forward network, each with residual connections and normalization. The decoder consisted of three sub-layers, one of which was a multi-head attention layer that focused on all outputs from the encoder. Many of these architectural components were modified in subsequent transformer model versions that followed the original.

Embeddings

In order NLP models to be able to process textual data, each word or token from the texts has to be converted to numerical representation, which is called embedding. Each token is assigned a unique vector that is supposed to capture the semantic meaning and context.

Positional Encoding

Positional encoding provides the model with information about the order or position of words in a sequence. Transformers lack an inherent notion of order or position because they process words in parallel rather than sequentially, so this is addressed by the positional encoding.

Multi-Head Self-Attention Mechanism

An attention function represents a mapping between a query and a set of key-value pairs. The multi-head attention in the transformer architecture uses a self-attention mechanism. All tokens in the attention layer's input are transformed into a query, key, and value by using three matrices. When a query and a key match well, indicating a high attention score, the corresponding value is highlighted in the output. The operation used to generate the output is scaled dot product matrix multiplication of the queries and keys. The resulting matrix demonstrates the degree of significance each word should place on other words.

Multi-head attention refers to multiple attention blocks that independently compute representations for each token. All of these representations are then combined to create the final representation of the token. The multi-head self-attention layer in the decoder is slightly different from the one in the encoder. It masks all tokens to the right of the token whose representation is being computed, ensuring that the decoder can only make use of tokens that come before the token it is attempting to predict.

An important feature of the multi-head self-attention mechanism in transformers is that each token is processed independently, enabling parallel computation of the representations for all tokens. Following that, transformers have two key advantages over RNNs: reduced computational complexity and enhanced connectivity, which is very beneficial for capturing long-term dependencies in sequences.

Generating Output Probabilities

After passing through all of the decoder blocks, the data goes through a final linear layer that functions as a classifier. Its size matches the total number of classes included, which is the number of words in the vocabulary. A softmax function is then used to generate the probability distribution over the target vocabulary, resulting in the final output sequence. The highest probability score points to the word that the model will predict as the next in the sequence.

1.4.2 T5

The process of pre-training a model on a large general-purpose dataset and then fine-tuning it for performing a downstream task is called transfer learning. The authors of the T5 model [55] explored a couple of transfer

learning methods for NLP by introducing a framework that converts all text-based language problems into "text-to-text" format. Their plan was to approach every text processing problem by taking text as input and producing new text as output. That's where the T5 model got its long name, "Text-to-Text Transfer Transformer." Their proposed framework made it possible to reuse the same model, objective, training approach, and decoding process for a variety of downstream tasks, including question answering, document summarization, and sentiment classification.

The creators of the T5 model constructed a new dataset called "Colossal Clean Crawled Corpus" (C4), which contains hundreds of gigabytes of clean English text scraped from the web. The C4 dataset was created using the publicly available Common Crawl web archive, which provides "web extracted text" by removing markup and other non-text parts from scraped HTML files. Because a large portion of this content was not natural language but rather gibberish text, they had to preprocess and clean it. They used a variety of preprocessing techniques, including removing citation markers, deduplicating the data, detecting and removing non-English texts, etc.

The T5 model architecture closely resembles the original encoder-decoder transformer implementation form [3]. A few changes they made to the original architecture included removing the layer norm bias, relocating the layer normalization outside the residual path, and adopting a different position embedding strategy. They used a simplified version of layer normalization in which the activations are rescaled but no additive bias is added. The original transformer architecture used a sinusoidal position signal or learned position embeddings, whereas they use relative position embeddings. Instead of using a fixed embedding for each position, relative position embeddings generate a different learned embedding based on the offset between the "key" and "query" being compared in the self-attention mechanism. They employed a simplified version of position embeddings, in which each embedding is essentially a scalar added to the associated logit used to compute attention weights. For efficiency, they shared the position embedding parameters across all layers in their model, although each attention head in a particular layer uses a different learned position embedding.

1.4.3 GPT-2

GPT-2 [54] is the second version of the Generative Pre-Training Transformer (GPT) models. It was released in 2019 by OpenAI. The authors wanted to demonstrate that language models can perform downstream tasks in a zero-shot setting with no parameter or architecture modifications. The GPT-2 model is based on a decoder-only architecture and is pre-trained for next-word prediction. These types of models only use the decoder during pre-training. They are also known as auto-regressive language models since they are taught to predict the next token based on the preceding sequence of tokens. They are most appropriate for text-generation tasks.

The GPT-2 model architecture is almost identical to the one suggested for GPT [59], with only a few modifications. The layer-normalization layer has been moved to the input of each sub-block. After the final self-attention block, an additional layer of normalization was introduced. The GPT-2 model can use 1024 input tokens and has a total vocabulary size of 50,257 tokens. The dataset used to train the GPT-2 models was WebText. It was created by crawling all Reddit links with at least 3 karma points. Following pre-processing, which included some de-duplication and heuristic-based cleaning, the dataset included a little more than 8 million documents, totaling 40 GB of text.

1.4.4 BioGPT

BioGPT [4] is a domain-specific, generative, pre-trained Transformer language model for biomedical text generation. The authors made use of the GPT-2 model architecture [54] and pre-trained the model on 15 million PubMed abstracts from scratch. The BioGPT model can be used for generating and mining text within biomedical literature.

The authors pre-trained the BioGPT model from scratch using data obtained from PubMed, using items updated before 2021 and excluding any that lacked abstracts. Instead of using the GPT-2 vocabulary, they built a custom vocabulary based on the in-domain corpus, that had a final size of 42,384 tokens. The model's backbone network is based on the GPT-2 model of medium size, with 24 layers, a hidden size of 1024, and 16 attention heads, totaling 355 million parameters. BioGPT itself has 347 million parameters, with the difference attributed to variations in embedding and output projection sizes due to the custom vocabulary[‡].

1.5 Text Generation Tasks and Decoding Strategies

The text generation tasks can be divided into two categories: directed and open-ended generation [60]. Directed generation tasks involve outputs that are constrained by and closely follow the structure or content of the input, ensuring the result remains aligned with the original information. This group includes machine translation and summarization. Open-ended generation encompasses conditional story generation and contextual text continuation. In open-ended generation, although the input context limits the range of acceptable outputs, there is still significant flexibility compared to the directed generation tasks. According to [60], certain tasks may fall somewhere between open-ended and directed generation. We believe that our task of creating descriptions for protein families, domains, and sequence features falls into this middle ground. Although it is essential for our summaries to be accurate and free from hallucinations, our models sometimes require a degree of freedom to include relevant information not explicitly stated in the input.

According to recent research, the decoding strategy used to generate text from the model may be more important than the model's architecture. Studies have shown that when a probabilistic neural text generator is trained with a maximum-likelihood objective, the most likely output is neither human-like nor high-quality [61]. As a result, a number of decoding strategies have been proposed, each claiming to produce more refined and desirable text than alternative methods.

There are two types of algorithms: deterministic and stochastic. Deterministic algorithms include greedy search, beam search, and its variants. Stochastic approaches, on the other hand, include ancestral sampling, top-k sampling, and top-p (nucleus) sampling [60]. Some of them will be discussed in further detail later. In their study, the authors of [62] compared deterministic and stochastic decoding strategies across various tasks. They found that directed generation tasks generally favor mode-seeking approaches, though this preference varies depending on the level of semantic constraint in the task. Specifically, as tasks become more semantically constrained, the preference for mode-seeking strategies grows stronger. For example, in open-ended tasks like story generation, they noted that even the best output from deterministic strategies ranked lower than the least favorable output from stochastic approaches, suggesting that mode-seeking strategies are unsuitable for such tasks. In contrast, for tasks

[‡]https://huggingface.co/microsoft/biogpt

like machine translation, mode-seeking methods work well. It can be concluded that the effectiveness of different decoding strategies does not universally apply across natural language generation tasks. For instance, while beam search and other mode-seeking approaches excel in directed generation tasks, they often produce repetitive or incoherent text in open-ended tasks. Specifically, in story generation, mode-seeking methods have led to significant text degeneration [62].

1.6 Evaluation of NLG Systems

Abstractive summarization (AS) is the task of shortening a document to a summary using new phrases that accurately and concisely capture the original document's contents. The summary should be concise, fluent, consistent with the source, and highlight the most important details. Certain summarization models have a significant drawback in that they produce summaries that are not factually consistent with the original document. These inconsistencies vary greatly and may include negations, incorrect entity use (for example, swapping subjects and objects), or hallucinations. As a result, detecting summary inconsistencies has become an active area of research.

An effective evaluation metric for the AS task should be capable of detecting the following:

- Summaries that have significant word overlap with the source or reference summary but contain factual errors.
- Summaries that are factually correct but missing important information.
- Summaries that are factually correct, provide enough information, and may differ in wording from the reference summary.

1.6.1 Human Evaluation

The best way to evaluate an NLG system is to have humans analyze its outputs. Evaluating NLG systems is a complex task that necessitates simultaneous assessment of an extensive list of qualities. In most AS evaluations, human evaluators are presented with the candidate summary as well as the source document or a list of references. Evaluators are often asked to score fluency, informativeness, lack of redundancy, referential clarity, structure, and coherence [63].

Some studies that use biological data request that human reviewers evaluate the subject using specific evaluation criteria. One specification developed by the authors of this paper [64] was for the evaluators to determine whether the summaries contained the correct or incorrect protein names. They also directed the evaluators to determine whether the resulting summary included all important protein functions, a subset of them, incorrect protein functions, and so on.

Human evaluations of text generation tasks are thorough but expensive, often taking months to complete. Also, they are involving human labor that cannot be reused. Automatic text evaluation reduces the need of expensive, time-consuming human evaluations, particularly when evaluating long, multi-sentence texts. Automatic metrics allow for faster evaluation during model training and testing, which makes faster the development of text generation systems. Many automated evaluation methods aim to capture similar quality characteristics that human evaluators would consider.

1.6.2 Reference-based Metrics for Automatic Evaluation

Reference-based metrics are used to evaluate generated text by comparing it to a reference, which is the human-annotated ground truth. Word-matching metrics such as ROUGE [65], BLEU [66], and others primarily rely on n-gram overlap comparisons and are widely used due to their computational efficiency. However, these metrics often struggle to perform well when information in the reference text is rephrased or reordered, as highlighted by [67]. They have also been shown to have a low correlation with human judgments [68].

That is why we chose to use methods that consider the semantic context. These methods are addressing the limitations seen in traditional metrics like BLEU and ROUGE. Several of them have been proven to have a high correlation with human judgments [69]. Examples of metrics using contextualized embeddings include BERTScore [69], Sentence Mover Similarity [70], and so on. They all use contextualized embeddings to determine the similarity between two texts. The semantic similarity of two sentences indicates how closely connected their meanings are. Firstly, each string is represented as a feature vector capturing its semantics. To quantify similarity between two embedding vectors, a common technique is to use cosine similarity. This metric ranges from -1 to 1 and represents the cosine of the angle between two non-zero vectors. The value 1 denotes that the two vectors are the same, whereas -1 indicates that they are distinct.

Limitations of the Reference-based Metrics

While these metrics are relatively simple, fast, and cost-effective to compute, they are still not "perfect" and lack the reliability that human evaluation provides. Studies [71] have identified several limitations, such as:

- **Lack of interpretability:** Unlike human evaluators, who assess multiple criteria (fluency, adequacy, coherence, relevance, thoroughness, etc.) for a given hypothesis, automatic evaluation metrics generate a single score. This single score often lacks clarity regarding which aspects it reflects, making it difficult to interpret.
- **Inherent bias:** Some metrics may exhibit biases; for instance, BERTScore can reflect societal biases [72].
- **Limited adaptability across tasks:** Evaluation criteria for NLG can vary significantly in different tasks, meaning that a metric suitable for one task may not effectively apply to another.
- **Inability to capture language nuances:** Even metrics designed specifically for a given task sometimes can't capture all the subtle details of that task. For example, the automatic evaluation metrics for AS have been critiqued because they don't check for factual inconsistencies in the summaries [73].

1.6.3 Reference-free Metrics for Automatic Evaluation

In some cases, obtaining reference texts can be time-consuming and costly, or they may be completely unavailable. This also applies to the descriptions of the IDRs that our model is supposed to generate. Ground truth descriptions for them cannot be obtained. Therefore, we had to rely on reference-free metrics for their evaluation.

Reference-free (context-based) metrics provide a score for the generated text without needing a ground truth. The evaluation is based on the context or source document instead. These methods are generally newer than

reference-based approaches, driven by the increasing need for scalable text evaluation. Some examples of reference-free metrics include the SummaC benchmark [74], Dependency Arc Entailment [75], QAFactEval [76], QuestEval [77], and others.

Limitations of the Reference-free Metrics

Although reference-free measures have been found to correlate well with human judgments, their use as the only way to evaluate some tasks has limitations. The authors of this study [78] claim that reference-free metrics are inherently biased and not fully capable of evaluating generated text. They argue that these metrics should not be used to evaluate tasks such as machine translation or summarization. Their explanation was that using reference-free metrics is like evaluating one generation model against another, which has several drawbacks:

- The metrics can be optimized at test time to find the best possible output.
- They tend to favor models that are more similar to themselves.
- They may also be biased against higher-quality outputs, including those created by humans.

As a result, they propose that reference-free metrics be used as diagnostic tools to understand and analyze model behavior rather than measuring a model's performance with the goal of achieving the highest possible score. When references are not available, they recommend allocating resources to gathering human-written references for evaluation rather than relying on reference-free metrics.

While the authors believe that using reference-free metrics to measure system quality is flawed, they do not advocate abandoning them entirely. Some aspects of generated text, such as the consistency of a summary to its input, can be assessed without the use of a reference [78].

1.7 Related Works

There are plenty of AI-based models that utilize protein data in order to perform various type of tasks. At this moment, the protein language models (pLMs) are gaining big popularity among the scientists. They have emerged as potent tools for predicting and designing protein structure and function. Anyways, most of them are not able to produce natural language, but usually they are used to produce either new protein sequences[79] or predicting protein function by predicting the GO terms associated with some protein[80].

ProtLLM is one model that caught our interest because it incorporates both structured protein data and textual data, such as biological research publications[58]. It is a versatile cross-modal LLM that is suitable for both protein-centric and protein-language tasks. ProtLLM has a unique dynamic protein mounting technique that allows it to handle complicated inputs containing plain language text interleaved with an arbitrary number of proteins. By creating a customized protein vocabulary, the authors empowered the model to predict not just natural language but also proteins from a large pool of candidates. In addition, they have created a large-scale interleaved protein-text dataset called InterPT for pre-training. This dataset includes both structured data sources, such as protein annotations, and unstructured data sources, like biological research papers. Their model has shown great

success on protein-centric tasks compared to the baselines and has induced zero-shot and in-context learning capabilities [58]. However, we did not use ProtLLM in our study because its GitHub repository showed ongoing development tasks, such as updating the model's version in HuggingFace [81].

The pre-trained models can be fine-tuned for a variety of downstream tasks, including classification, question answering, dependency parsing, named entity recognition, and more. In our case, we are mostly interested in applying LLMs for the summarization downstream task, and a few examples from the biological domain will be further discussed. KeBioSum has been described as knowledge infusion training framework. Its authors have used generative and discriminative training techniques to create knowledge adapters and have injected them into the pre-trained language models to fine-tune them for the extractive summarization task [82]. BioBERTSum uses a domain-aware bidirectional language model pre-trained on large-scale biomedical corpora to capture token-level and sentence-level contextual representation. The language model is then fine-tuned for extractive text summarization on a single biomedical document. The authors use a sentence position embedding mechanism that allows the model to learn sentence position and document structure [83].

GO2Sum is a fine-tuned version of the T5 model created for summarizing GO term descriptions into function, subunit, and pathway paragraphs for the UniProt database entries. When compared to the results obtained using the pre-trained T5 model, GO2Sum's summaries have much better resembled the ground truth UniProt data. The model has been evaluated by using embedding-based scoring methods, and the results showed a strong correlation with human evaluators' assessments. While GO2Sum produced accurate summaries in general, a few notable limitations have been reported. These include its inability to mention specific protein names and the maximum input token size of 1024, which has not been sufficient for some longer inputs. To resolve the first limitation, the authors propose adding additional information sources, such as protein-protein interaction data. For the second issue, they believe that employing GO term pruning or text compression may be a solution. The aim of this work had been to improve the interaction between humans and machine learning models, particularly in terms of supporting biologists and medical scientists in their daily research actions [64].

The ProtNLM model is another work that makes use of protein data, and the results are already accessible on the UniProt knowledge base website. According to the authors of this preprint [84], one of the most difficult challenges in the biological sciences is predicting a protein's functional characteristics directly from its sequence and thus discovering new proteins with specific functions. This requires the model to create an association between amino acid sequences and natural language. As a first step, they have trained models for predicting free text natural language captions that describe amino acid sequences' functional properties. A set of metrics for evaluating the model performance has been created, and curators from the Pfam [85] and UniProt databases manually reviewed the model predictions that did not match existing functional annotations. The feedback has been used to further improve the model. The resulting model, ProtNLM, has been used to predict protein names for approximately 49 million previously uncharacterized proteins, and the results are now part of the UniProt database [84]. On Figure 1.7 [§] can be observed one example of protein that has artificially generated name by the ProtNLM model.

In addition to UniProt, the InterPro database uses AI-generated texts to address the lack of annotations for certain families. They have decided to use OpenAI's GPT-4 LLM for generating the summaries. An approach that involves extracting description lines and function comments from Swiss-Prot entries in the family and aggregating them into a prompt that is sent to the OpenAI API has been employed. It has been concluded that when there

[§]Retrieved from: https://www.uniprot.org/uniprotkb/A0A2Z4IEP2/entry

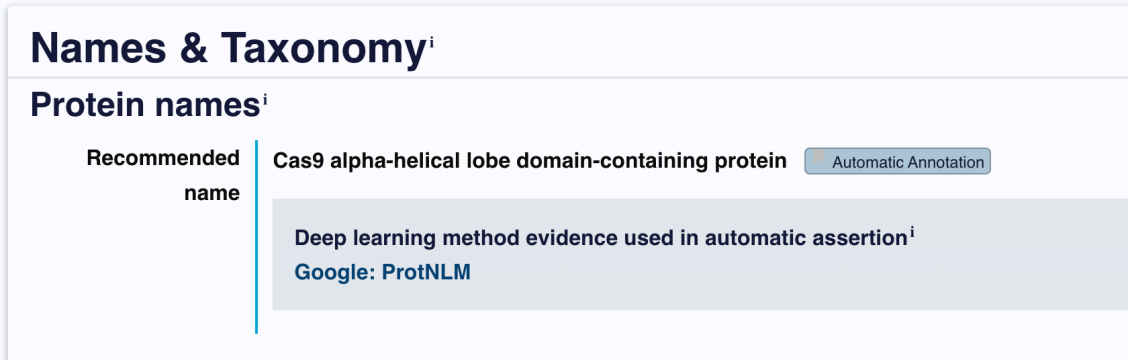

Figure 1.7: Example of protein name generated by ProtNLM in UniProtKB.

was sufficient information from Swiss-Prot, the GPT-4 [50] model was able to generate reasonably high-quality summaries. Since the release of InterPro 97.0, descriptions for around 5,500 unintegrated PANTHER [86] families have been successfully created. Exploring the usage of LLMs for generating summaries for families for other member databases and different InterPro entries has been mentioned as a future plan. Figure 1.8 [¶] shows example of entry on the InterPro website that has AI-generated text associated with it. It is worth noting that it has specific tags that distinguish it from the entries with human-written texts.

1.8 Research Contributions

Our primary goal was to develop an NLG model capable of generating descriptions for IDRs. As previously mentioned, these brief descriptions are essential for researchers to gain quick insights into the functional roles of IDRs. The DisProt database contains expert-curated information on IDRs and IDPs, annotated with appropriate GO and IDPO terms based on findings from various research papers. However, it lacks human-written short descriptions for its entries. Consequently, we were unable to collect any IDR-related ground truth data that would serve for fine-tuning our models.

To address this, we decided to fine-tune the models with short descriptions of other protein sequence signatures, such as protein domains, families, and sequence features, which are available in the InterPro database as ground truth data. While we acknowledge that the format and content of these descriptions might not perfectly align with expectations for IDRs, our goal was to evaluate the model's learning capabilities and its potential to apply this knowledge to generating descriptions for IDRs. The main contributions of this study are as follows:

- **Datasets Construction:** We constructed three datasets of varying complexity using data from the InterPro database, designed to train models to generate descriptions for protein sequence signatures such as domains, families, and sequence features. Each dataset employs a structured input template to organize textual information effectively. Additionally, we created a dedicated dataset for intrinsically disordered regions (IDRs), using the same input template, to enable the generation of concise descriptions for IDRs based on data from the DisProt database.

[¶] Retrieved from: https://www.ebi.ac.uk/interpro/entry/panther/PTHR18806/

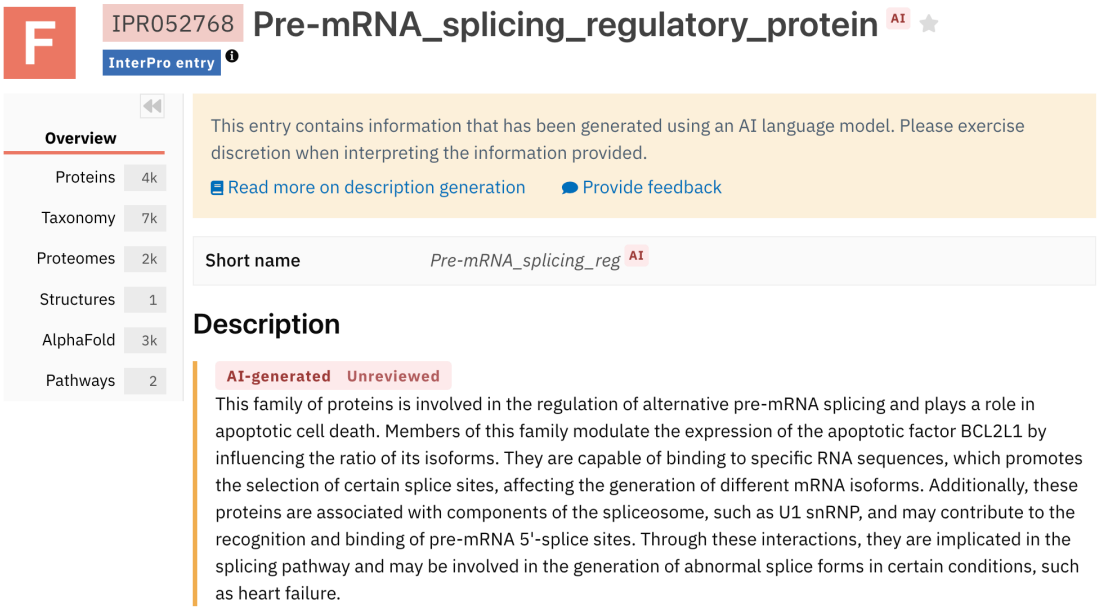

Figure 1.8: Example of InterPro family entry that has AI-generated name, short name and description.

- **Evaluating the Impact of Dataset Complexity:** We fine-tuned the pre-trained T5 model using three datasets of varying complexity and then evaluated the models' performance on the test set. By comparing the performance of these models, we were able to determine how increasing dataset complexity affected the quality of the generated summaries.
- **Comparison of Pre-Trained and Fine-Tuned T5 Models:** We compared the performance of pre-trained and fine-tuned T5 models to gain insights into the effectiveness of task-specific training.
- **Comparative Analysis of Fine-Tuned T5, GPT-2, and BioGPT Models:** We performed a comparative analysis of summaries generated by the fine-tuned T5, GPT-2, and BioGPT models. The summaries were generated using beam search and sampling as decoding strategies. Three reference-based evaluation metrics were used to assess their performance.
- **Generating Descriptions for IDRs:** One of the best-performing models was used to generate short descriptions for IDRs, which were afterwards evaluated with a reference-free evaluation metric. We discussed its adaptability, common mistakes, and areas for improvement.

This work bridges the gap between machine-readable ontologies and human-readable short descriptions, providing a foundation for automated IDR short description generation. More information about these contributions will be presented in the following two chapters, "Materials and Methods" and "Results."

2

Materials and Methods

In this chapter, we will first describe the process of dataset construction and preprocessing. Afterwards, we will go over the methods that we used to create and evaluate our models for generating protein signature summaries in more depth. The concept of fine-tuning LLMs will be discussed, followed by some specifics on the hyperparameters we used to fine-tune our three models. We will then outline the decoding strategies that we used, and the hyperparameters configured for text generation to optimize output quality. Finally, the evaluation metrics we employed will be presented.

2.1 Data Preparation

In the field of AI, the performance of models trained to solve specific problems is frequently evaluated and compared using benchmarks. A benchmark consists of a task (e.g., summarization), a dataset, and metrics for evaluating model performance. Benchmark datasets provide researchers with a standardized method for evaluating and comparing model performance, and they are typically publicly available. We are unaware of any benchmark dataset for the specific problem addressed in this study. Consequently, we created custom datasets designed to meet the needs of our research. In the following paragraphs, we will describe our input and output features in detail, followed by the preprocessing steps and the splitting of the datasets into training and testing subsets.

2.1.1 Ground Truth Selection

As stated in the introduction to the thesis, our goal is to teach the NLG model to generate descriptions of protein domains, families, and sequence features. Therefore, we decided to use the short descriptions for all InterPro entries as our output features (summaries). They will serve us as ground truth when training and evaluating our

models. We obtained them from the "interpro.xml.gz" file*, by parsing it and extracting the descriptions for every entry (May, 2024). Figure 2.1† shows one example of a description from our dataset available on the InterPro website. The preprocessing of summaries involved removing non-textual elements, such as references to related papers.

2.1.2 Input Templates Design

The next step involved defining the input texts that the model would use to generate summaries, which proved to be the most challenging part. This was because the ground truth summaries have been carefully curated by expert biological curators. These curators probably combined information from multiple databases, relevant research papers, and their own biological expertise to write the summaries. We drew inspiration from the [64] approach, in which GO terms descriptions have been used for generating summaries of protein functions. Similarly, our initial idea was to use the GO terms descriptions associated with each InterPro entry as input for our model. The first dataset that we created, which is the least complex of our datasets, will be referred to as the "GOTerms" dataset in the following paragraphs.

To create the dataset, we first obtained the GO term IDs linked to each InterPro entry from the file named "interpro2go"‡ (May, 2024). It contained mappings of InterPro entries to GO terms. We only included the InterPro entries that had at least one associated GO Term in our dataset. For each GO term, we then retrieved the corresponding text description from the Gene Ontology Consortium site§ (June, 2024).

Furthermore, it was necessary to combine and organize the textual data from multiple GO terms into structured input that could be processed by our summarization model. In order to achieve that, we created a template for the input, as shown in Figure 2.2. The template starts with an introductory sentence that identifies the type of biological entity we are generating a description for. This entity type can be a domain, family, homologous superfamily, conserved site, active site, repeat, binding site, or PTM. Following that, the name, description, and namespace of each GO term associated with the InterPro entry are listed, with each element separated by a pipe symbol ("|"). The distinct GO terms are placed in different lines, i.e., they are separated by a new line separator. An example of an input constructed using this template, along with its corresponding output, is shown in Figure 2.3.

GO Terms Count Distribution

We analyzed the distribution of the number of GO terms associated with the InterPro entries in our dataset. Figure 2.4 displays the resulting histogram. The x-axis represents the number of GO terms linked to each entry, while the y-axis shows the frequency of entries for each GO term count. As can be seen, the majority of InterPro entries have a relatively small number of GO terms linked to them, with over 4,000 entries associated with only 1 GO term. Notably, no entry has more than 11 GO terms. This limited number of functional annotations created instances where the input texts were shorter than the outputs the model was expected to generate. To address this

*The file can be downloaded from here: https://ftp.ebi.ac.uk/pub/databases/interpro/releases/100.0/
†Retrieved from: https://www.ebi.ac.uk/interpro/entry/InterPro/IPR000294/
‡The file can be downloaded from here: https://www.ebi.ac.uk/interpro/download/
§Website URL: https://geneontology.org/docs/download-ontology/

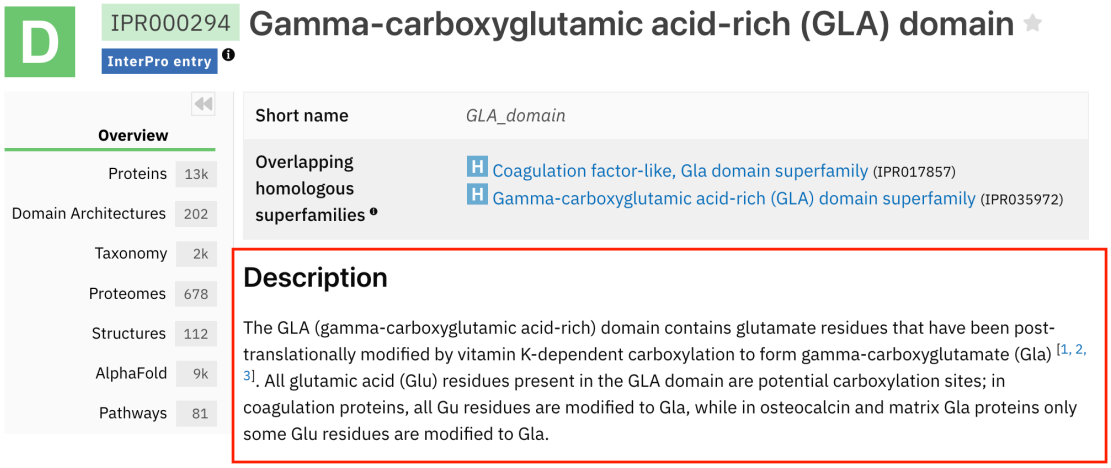

Figure 2.1: Example of the description for one InterPro entry (Gamma-carboxyglutamic acid-rich (GLA) domain) marked with a red rectangle

issue and enhance the dataset, we decided to propagate the GO terms and see if we could get a better-performing model with the enriched dataset. The concept of propagating GO terms will be explained in the next paragraph.

GO Terms Annotations Propagation

The definitions of what is gene ontology and gene ontology terms have already been presented in the introduction chapter. Here we are going to explain shortly why propagating the GO term labels through their ancestors up to the root is justified and how it was performed. When a gene is associated with some GO term, it is automatically linked to its parent terms in the hierarchy as well. Since GO annotations inherit all the attributes of their parent terms, every pathway from a term back to its root must be biologically accurate; otherwise, the ontology needs revision [87]. In our propagation process, the relationships "is a," "part of," and "regulates" between terms were considered. The propagation was carried out using the GOATOOLS Python package [88]. The same input template used for the "GOTerms" dataset was applied to this dataset as well. From this point forward, we will refer to this dataset as the "PropGOTerms" dataset in the following sections.

Protein Function Descriptions Enrichment

The third and most complex dataset was generated using not only GO term information but also functional descriptions from the literature, as recorded in the UniProtKB database, for proteins associated with specific InterPro signatures. Additionally, the dataset includes information about the corresponding protein organisms. The dataset will be referred to as the "GOTerms + SwissProt" dataset from now on. Figure 2.5 illustrates the template for this dataset. The first section of the template mirrors that of the "GOTerms dataset," while the second part provides functional descriptions for each reviewed protein from the Swiss-Prot database, along with the associated organism names.

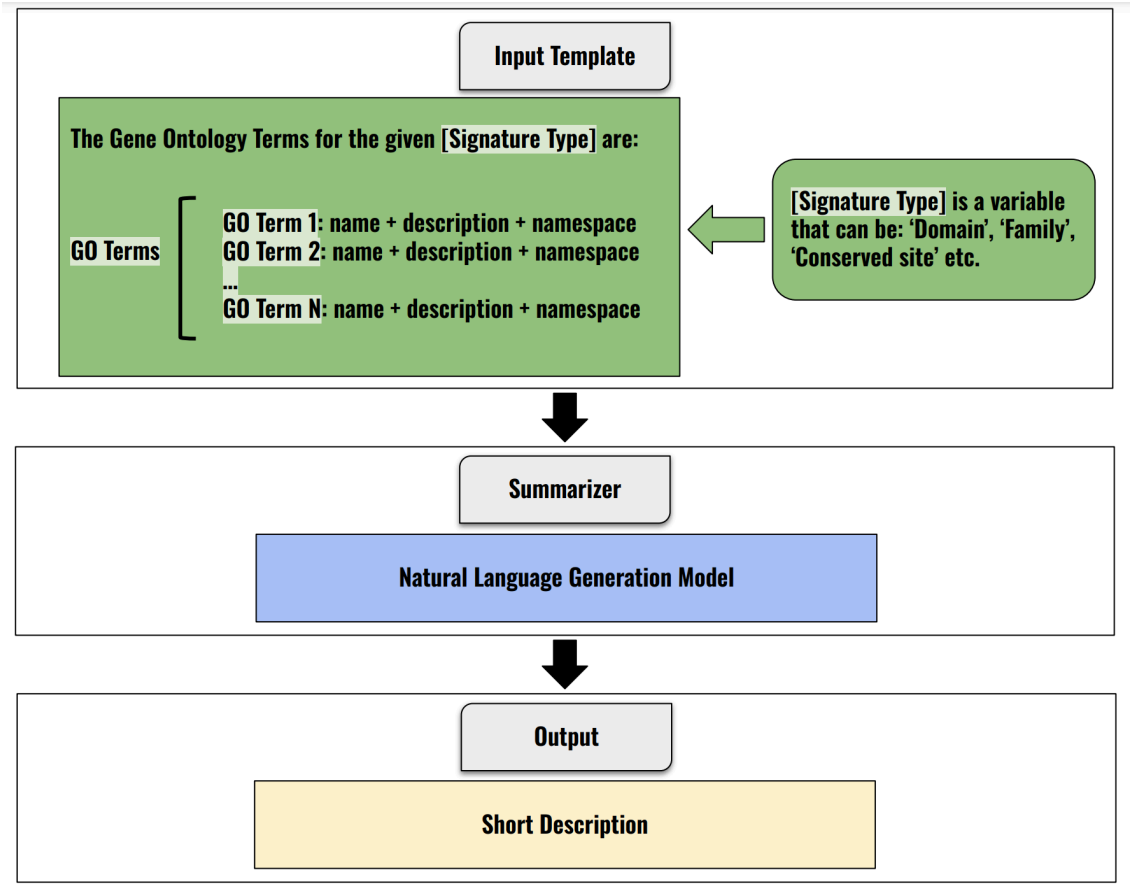

Figure 2.2: Workflow of the summarization with the "GOTerms" dataset.

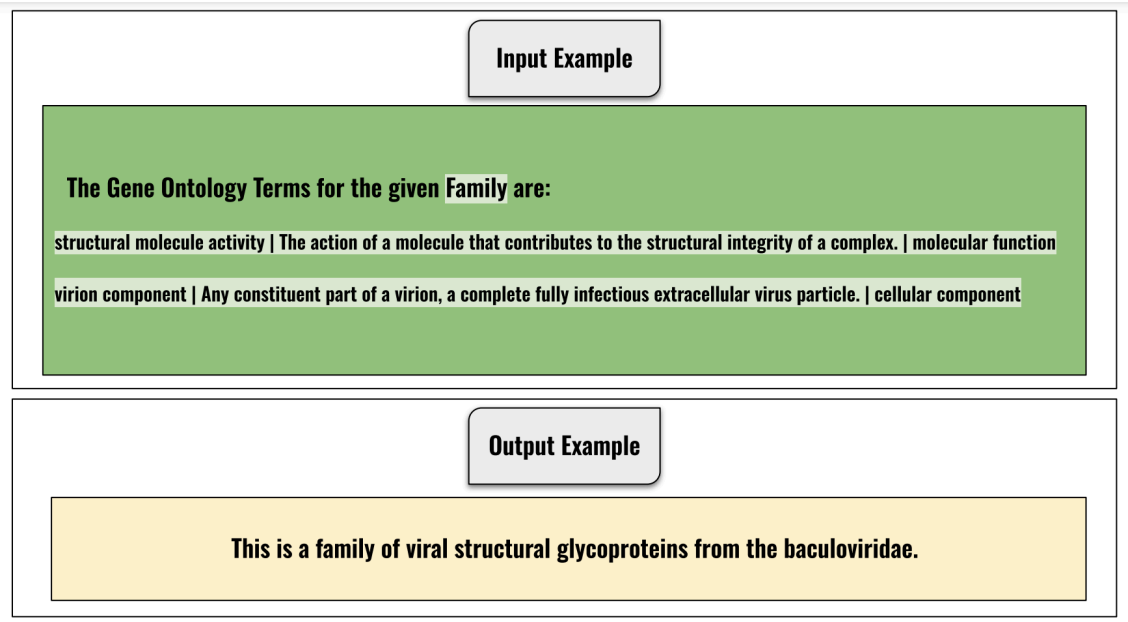

Figure 2.3: Example of one input and output pair from our "GOTerms" dataset (InterPro family entry with IPR006790 ID).

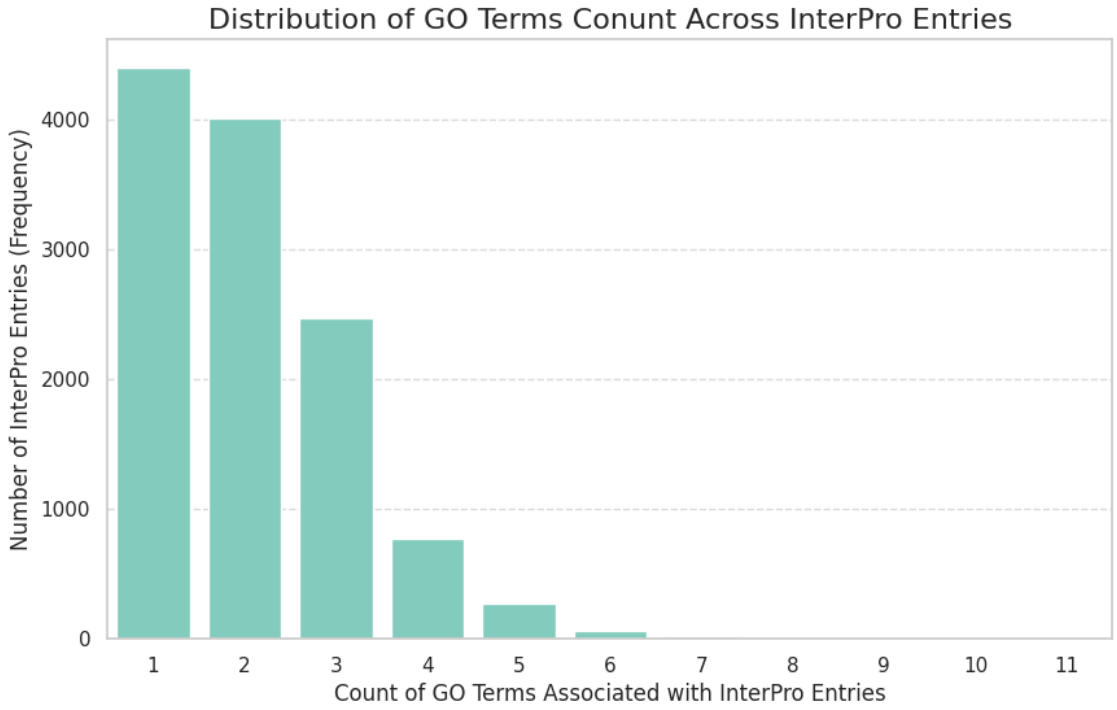

Figure 2.4: Plot of the counts of GO terms associated with InterPro entries from the dataset.

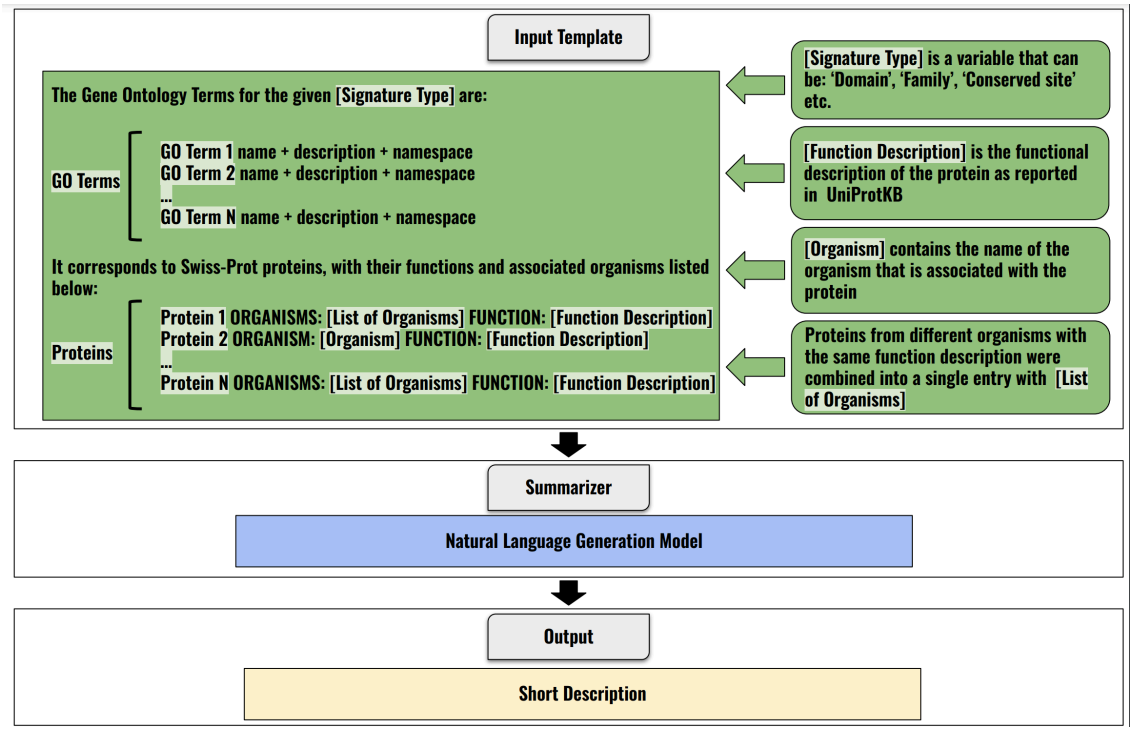

Figure 2.5: Workflow of the summarization with the "GOTerms + SwissProt" dataset.

In some cases, multiple proteins shared the same functional description. For example, the proteins with the following UniProtKB IDs: "P0DOK4," "W6JGV7," and "A0A1P8YT89", have identical functional descriptions but are found in different organisms. To avoid redundancy, we combined the organism names into a list (limited to five organisms per list to manage instances with many organisms) and included the functional description only once. This approach minimized repetition in the dataset. Figure 2.6 provides an example of one sample from this dataset.

Protein IDs associated with each InterPro entry were sourced from UniProtKB[¶]. Only the reviewed (Swiss-Prot) proteins have been considered, and the InterPro cross-reference column was selected to retrieve the mapping to the InterPro entries. The short protein descriptions were also obtained from the same source.

2.1.3 Data Preprocessing

We excluded proteins without functional descriptions from the dataset and retained only those with non-empty descriptions. Additionally, as part of the preprocessing, all references to papers were removed from the protein descriptions (e.g. {ECO:0000269|PubMed:29395922}). This was done to ensure the dataset contained only plain text, as the references could only confuse the model. We are not providing it with access to those papers that are referenced, so that is why having references was redundant.

[¶]https://www.uniprot.org/uniprotkb?query=reviewed:true

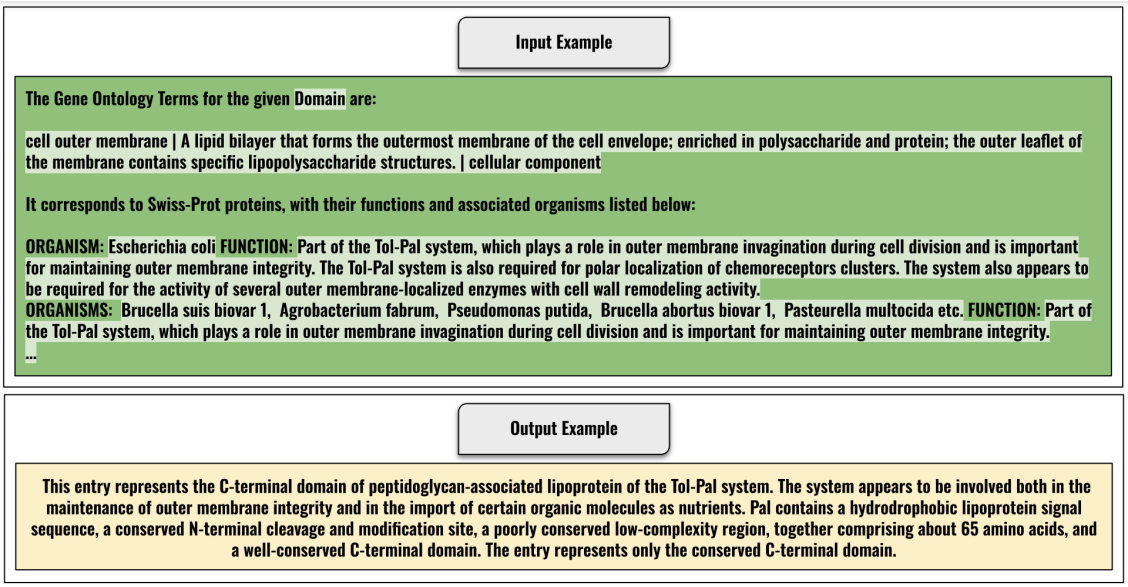

Figure 2.6: Example of one input and output pair from our "GOTerms + SwissProt" dataset (InterPro domain entry with IPR014169 ID).

Preprocessing was also applied to the organism names. In some cases, the names included overly specific details, such as strain information and other details that contained numbers in parentheses. Those have been removed from the organism names. For example, "Acinetobacter baumannii (strain 1295743)" was simplified to "Acinetobacter baumannii," which is referenced as its parent organism in the UniProt taxonomy. This simplification was necessary because the organisms were included to help the model distingish broader groups, such as eukaryotes, viruses, or plants. The detailed strain information was not relevant to the descriptions the model was supposed to generate.

Next, we checked for duplicate entries and removed them, reducing the dataset to 12,475 entries. We also eliminated extremely long inputs, as the pre-trained models that we intended to use for fine-tuning were not designed to handle texts of such length. Using the NLTK tokenizer [89], we counted the number of tokens in each entry and removed those with more than 5,000 tokens. After these adjustments, the final dataset contained 12,020 samples.

Once the dataset preprocessing was finalized, we plotted the number of dataset instances per InterPro signature type, as shown in Figure 2.7. The largest category was "Family," followed by "Domain." The remaining categories had fewer samples.

2.1.4 Dataset Splitting

When splitting the dataset into training and test subsets, we ensured that the same InterPro entries were used across all three dataset types ("GOTerms," "PropGOTerms," and "GOTerms + SwissProt"). Specifically, if an entry was included in the training set for one dataset type, it was also included in the training set for the other two dataset types, and similarly for the test set. This approach maintained consistency and allowed for a meaningful

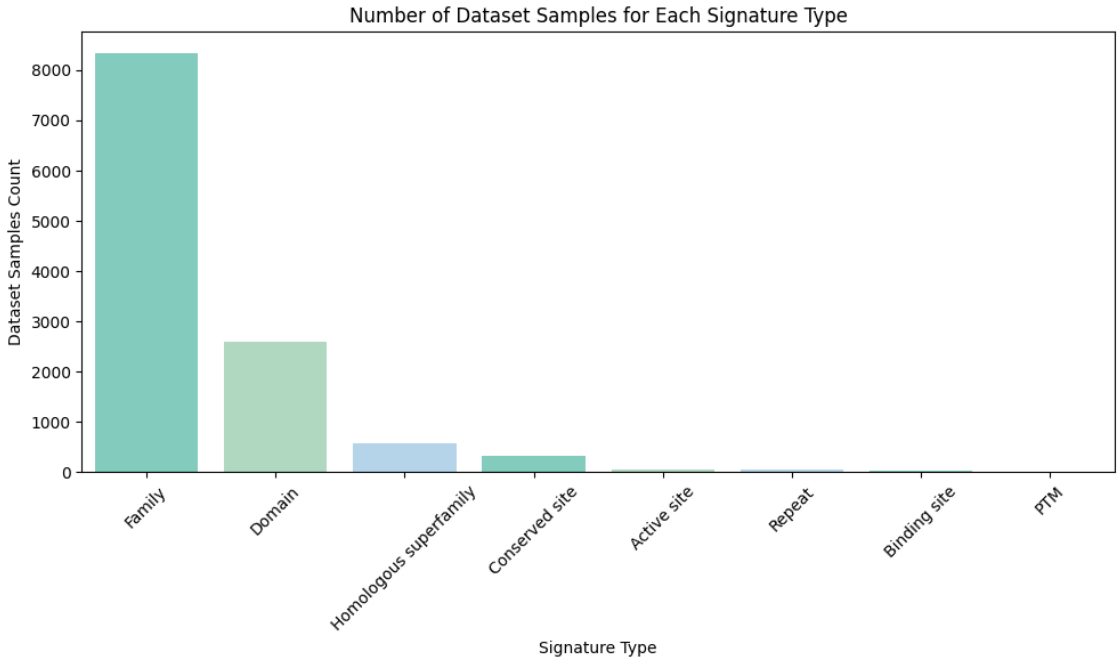

Figure 2.7: Number of samples from the dataset for each signature type

comparison between the models. The dataset was divided such that 80% of the samples were used for training and 20% for testing. Additionally, the test set was used for model validation during fine-tuning.

2.1.5 IDRs Dataset Creation

To create a dataset for IDRs that is consistent in format with our previous datasets, we used the "GO Terms + SwissProt" dataset input template with data from the DisProt database. First, we downloaded the June 2024 DisProt release from the DisProt database[‖]. Regions in this dataset were annotated with either GO or IDPO terms. We retained all GO terms and selected only IDPO terms from the "Disorder function" namespace, discarding those from the "Structural state" and "Structural transition" namespaces because they did not correspond to functional descriptions of interest.

Next, we created a unique ID column for each IDR by combining the protein ID with the start and end positions of the IDR. Rows with identical IDR IDs were merged, keeping multiple GO and IDPO terms in list format for each ID.

Initially, each IDR was associated with a single protein, so we performed a BLAST [90] search across the entire UniProt database to identify other proteins containing similar regions. Only BLAST results with more than 90% sequence identity overlap were kept.

We extracted protein functional descriptions and organism data from UniProt using SPARQL queries at their official endpoint ("https://sparql.uniprot.org/sparql"). Proteins with non-empty functional descriptions were

[‖]https://disprot.org/download

considered. The same preprocessing steps as in previous datasets were applied. Out of the initial 2,650 unique IDR IDs, we retained 2,420 entries because 230 regions lacked functional descriptions in any associated protein. For GO terms, descriptions were added in the same manner as for the previous datasets, and IDPO descriptions were obtained from the "IDPO_v0.3.0.obo" file downloaded from the DisProt database.

Finally, we inserted "Intrinsically Disordered Region" in the place of the "[Signature Type]" placeholder in each entry. The dataset will be referred to as the "IDRs" dataset from now on. An example dataset entry is provided in Figure 2.8.

2.2 Fine-Tuning of Large Language Models

Foundation (pre-trained) models are type of models that have been trained on broad and large data sets and can be adapted or fine-tuned to perform specific downstream tasks. At the technical level, their capabilities rely on transfer learning, a method in which models apply learned knowledge from one task to improve performance on another. Several factors affected the development of extremely powerful foundation models, including improvements in computer hardware, the advent of the transformer architecture that we discussed earlier, and vast volumes of training data. Nevertheless, obtaining large datasets that are labeled is always difficult, which is why the majority of these foundation models were trained with a self-supervised approach. The idea behind self-supervised learning is that the pre-training task is created autonomously from unannotated data. This enables models to learn while predicting sections of the inputs, making them more broad and perhaps of greater value than models trained on a limited label space [91].

There are many self-supervised pre-training tasks that can also be applied for fine-tuning the LLMs. One of these is the unidirectional Language Modeling (LM) task, which was used to develop both the GPT-2 and BioGPT models. The main objective of the LM task is to teach the model to predict the next token. The approach is very similar to Causal Language Modeling (CLM), which is based on predicting a text sequence auto-regressively, meaning that the model generates content based solely on the preceding words. Besides the basic LM, there are a few more variations of the LM task. One of them is the bidirectional LM technique based on predicting both the next and previous tokens. The Masked Language Modeling (MLM) task is also commonly used. It functions such that some tokens from the input are being masked and the model is trained to predict those tokens using the surrounding context. Another type is the Denoising Autoencoder (DAE), in which the model accepts a partially corrupted input and attempts to recover the original, undistorted input. Examples of corrupted input include masking arbitrarily picked elements from sequences, removing tokens at random, and mixing sentences in random order. The T5 model is pre-trained with a DAE task [92].

In the following paragraph, we will briefly discuss the library that we used for the technical implementation of our fine-tuning tasks. Afterwards, we will discuss the specifics of fine-tuning the three models used in our study. We will discuss the fine-tuning tasks, the hyperparameter values that we chose and the changes we made to the datasets before the fine-tuning process.

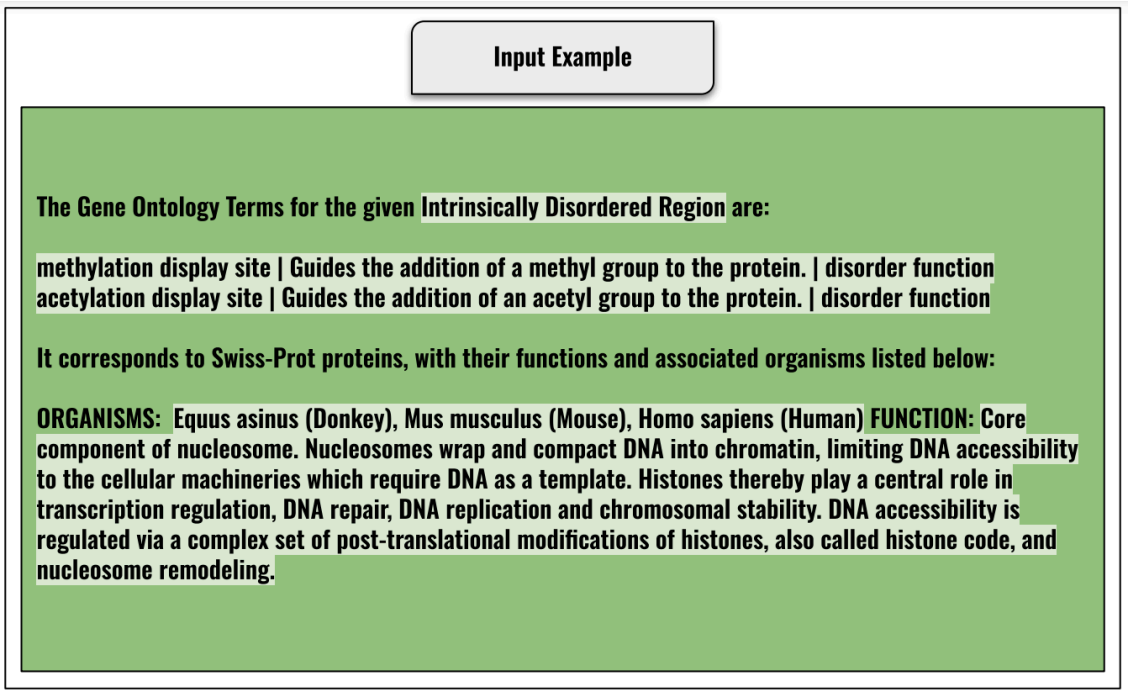

Figure 2.8: Example of one input from our "IDRs" dataset (DisProt entry with DP01157 ID, IDR spanning positions 1 to 27 in the protein sequence.).

2.2.1 HuggingFace Transformers Library

HuggingFace Transformers [81] is a library that supports transformer-based architectures and simplifies the distribution of pre-trained models. The library provides an implementation of the transformer architecture and enables access to a wide range of pre-trained models through a centralized model hub**. This hub allows users to compare multiple models using the same simple API and experiment with shared models on a wide range of tasks.

This library was created to simulate the typical NLP model pipeline. It offers tools for training and fine-tuning transformer models. Every model in the library is fully characterized by three building blocks:

- Tokenizer - which converts raw text to sparse index encodings.
- Transformer - which translates sparse indices to contextual embeddings.
- Head - which makes a task-specific prediction using contextual embeddings.

Tokenizers are specific to each model, handling encoding and decoding based on the model's requirements. They can be loaded from a pre-trained model or set up manually, and they store the vocabulary token-to-index map for their corresponding model. They can be customized by adding new tokens or modifying the model-specific special tokens. Transformer models can be paired with a variety of pre-implemented heads designed for common tasks. Some heads, such as those used for conditional generation, can also perform advanced functions

**https://huggingface.co/models

like sampling [93] and beam search [94]. For pre-trained models, the library includes the original heads used during pre-training. Additionally, the same core Transformer parameters can be used with different heads for fine-tuning on new tasks.

Hugging Face Transformers' Trainer class enables feature-complete training in PyTorch and allows distributed training across many GPUs/TPUs. The Seq2SeqTrainer class is derived from the Trainer class and is designed for training models for sequence-to-sequence tasks like summarization and translation. We fine-tuned the GPT-2 and BioGPT models using the Trainer class, and T5 with the Seq2SeqTrainer class. We will dive into more detail in the next paragraphs.

2.2.2 T5 Model Fine-Tuning

The authors of [55] produced T5 models in a variety of sizes, including small, base, and large. The one that we used for our study was T5-Base[††], a checkpoint with 220 million parameters.

The model was set up with pre-trained T5-base weights and fine-tuned for ten epochs. The Adam optimizer was used, and the learning rate was set to 0.00005. Training and evaluation were carried out with a batch size of four, and a seed of 42 was used for reproducibility. The input token length was set to 1024, with a target maximum token length of 256. At the end of each epoch, we ran an evaluation with our validation set, using the ROUGE metric. During fine-tuning, input sequences were prefixed with 'summarize: ' because that was recommended in order to specify the summarization task for the model.

2.2.3 GPT-2 Model Fine-Tuning

We used the GPT-2 version[‡‡] with 117 million parameters and 12 decoder-transformer blocks. The embedding size of each token in this version is 768 dimensions.

The model was initialized with the weights from the pre-trained GPT-2 model, and the fine-tuning task was CLM. We set a learning rate of 5e-05, with a training and evaluation batch size of 4, and the process was run for 10 epochs. The Adam optimizer was employed with specific parameters (betas=(0.9, 0.999) and epsilon=1e-08) and a linear learning rate scheduler. Additional special tokens were added to the tokenizer. The marker "<|in|>" indicated the beginning of the input texts, while "<|out|>" indicated the beginning of the outputs.

2.2.4 BioGPT Model Fine-Tuning

The authors of the BioGPT model carefully designed and analyzed the target sequence format and prompts to better model and adapt to downstream tasks. They have proposed how the source and target can be used to fine-tune and do inference with BioGPT. Instead of accomplishing this in a naive manner, i.e., concatenating the source and target sequences together, they advocated employing prompts, referring to the concept as prompt-based fine-tuning. Prompts can be used to add task-specific instructions to the input, allowing the model to provide output that is more suitable for that task.

[††]https://huggingface.co/google-t5/t5-base
[‡‡]https://huggingface.co/openai-community/gpt2

The final sequence format that they used for fine-tuning was [source; prompt; target]. For inference, they proposed using the source text and prompt as input to guide the language model, which then generates the target output. Following this approach, we shaped our dataset accordingly. Figure 2.9 illustrates our dataset format adapted for downstream tasks based on the BioGPT framework.

The hyperparameters were configured exactly as they were for fine-tuning the GPT-2 model. For BioGPT, we observed signs of overfitting after the 6th epoch, so we stopped the fine-tuning at that point and used this version of the model for evaluation and comparison with other models.

2.3 Decoding Strategies for Text Generation

In the introduction of the thesis we already explained the types of text generation tasks, and the different decoding strategies that can be used for them. Here we will describe in more depth the decoding strategies that we decided to use for our problem, and the hyperparameters values that we set.

The HuggingFace Transformers library supports many types of decoding strategies, such as greedy decoding, contrastive search, multinomial sampling, beam-search decoding, beam-search multinomial sampling, diverse beam-search decoding, constrained beam-search decoding, assisted decoding, and dola decoding. Different parameters can be configured to adjust the generating strategy used. We decided to use two decoding strategies for generating summaries with our fine-tuned models: beam-search decoding and multinomial sampling. We then compared the quality of the summaries generated by each strategy across all three models.

2.3.1 Beam-Search Decoding

Beam-search maintains multiple hypotheses at each time step, ultimately selecting the hypothesis with the highest overall probability for the complete sequence. This method can identify high-probability sequences that start with lower-probability initial tokens, which a greedy search would typically overlook.

Apart from choosing the generation technique, there are also additional parameters that enable users to manipulate the model output logits. These parameters can influence the quality and diversity of generated text. Some of them are:

- The **length penalty** parameter controls the length preference in beam-search decoding. When applied as an exponent to the sequence length, it affects the sequence score by penalizing or favoring particular lengths. A positive length penalty (>0) promotes longer sequences, while a negative penalty (<0) encourages shorter outputs. This allows for more precise control over output length to better match task requirements.
- The **no repeat n-gram size** parameter minimizes repetition by limiting the reuse of word sequences, or n-grams, of a certain length in generated text. When set to a value greater than zero, any n-gram of that length will only appear once in the output, improving fluency and readability by avoiding repetitive patterns.

We used the beam search visualizer tool (https://huggingface.co/spaces/m-ric/beam_search_visualizer) to create the graphical representation showed in Figure 2.10, which demonstrates how beam search decoding works. It can be seen on the image that there are step scores and total (cumulative) scores that are being calculated for

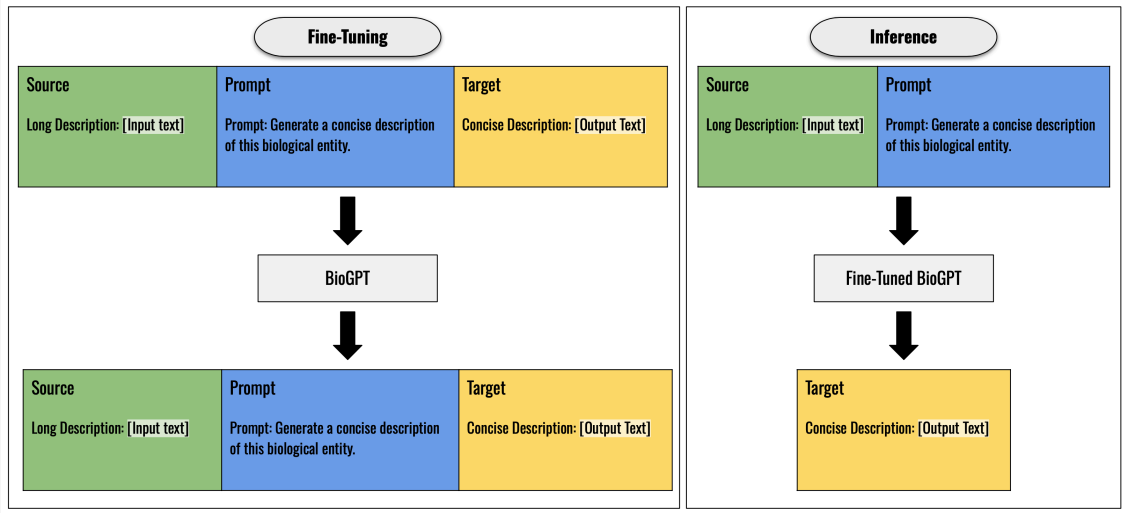

Figure 2.9: Dataset formatted according to the BioGPT framework proposed in [4] for adaptation to downstream tasks.

every token. The final score that is used for ranking the sequences at the end, in order to choose the best one, is calculated with the following equation:

$$\text{score} = \frac{\text{cumulative\_score}}{\text{output\_length}^{\text{length\_penalty}}} \tag{2.1}$$

It can be noticed that the length penalty has influence when calculating the final score. This final score represents the log probability of the sequence, and it is a negative value. The values closer to zero indicate higher probabilities and therefore better ranking. In the visual representation, the three top scoring sequences are highlighted in blue, while the lower-ranked sequence is highlighted in yellow. The highest ranked sequence is the following: "Proteins are one of the most important macromolecules found in all living things.", with a cumulative score of -0.79, which will be used for calculating the final score with the above-mentioned formula.

When applying the beam-search decoding strategy, we set the number of beams to four. Moreover, we set a length penalty of 2 to encourage the model to generate more complete, well-developed sequences. Additionally, the no repeat ngram size parameter was set to 3, in order to prevent the model from repeating trigrams within the output, enhancing output diversity. Early stopping was enabled to terminate generation as soon as the best beams reached a sufficiently high score, optimizing both efficiency and relevance of the generated text.

2.3.2 Sampling

The second strategy that we used, multinomial sampling, is a stochastic decoding strategy. Unlike greedy search, which always chooses the token with the highest probability as the next token, multinomial sampling (also known as ancestral sampling) selects the next token at random depending on the probability distribution over the entire vocabulary provided by the model. The model reduces the possibility of repetition by giving every token with a non-zero probability a chance to be selected. Some additional parameters that allow manipulating the model

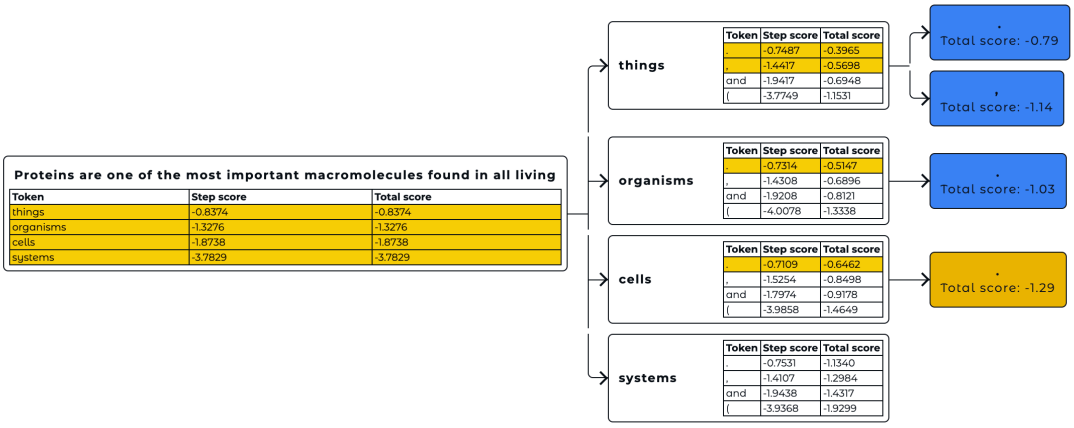

Figure 2.10: Example of beam-search decoding.

output logits while doing the sampling are:

- **Temperature** is a parameter in language models that controls the "creativity" of generated outputs by altering the distribution of token probabilities. When the temperature is high, the probability distribution flattens, allowing the model to select less likely but more diverse phrases, resulting in more creative and unexpected responses. Lower temperatures increase the likelihood of high-probability tokens, producing more focused and consistent outputs with fewer variations. Setting temperature to 1 maintains the original probability distribution, balancing between diversity and coherence.
- **Top-k** sampling is an approach for reducing the computational requirements of text generation by restricting the possible output tokens to only the k most probable alternatives. Instead of computing probabilities over the entire vocabulary, the model selects and applies softmax only to the top k highest-probability tokens. This strategy maintains a balance between efficiency and diversity, such that lower values of k produce more predictable and repetitive texts, while larger values allow for more variability and creativity by including less common tokens in the sampling process.
- **Top-p**, also known as nucleus sampling[60], is a strategy that allows the model to consider only the smallest set of most likely tokens whose cumulative probability is equal to or greater than a certain threshold. This allows the model to adjust the diversity of its responses to the context, improving appropriateness for specific prompts by focusing on relevant options. For example, a top-p value of 0.9 indicates that only tokens with probability equal to 90% or higher are included in the sampling pool. If top-p is set to one, the model includes all possible tokens, producing more diverse but potentially less focused outputs.

We initially tested the multinomial sampling technique using the HuggingFace library's default values for the parameters. The temperature was set to 1, top-k was set to 50, and top-p was set to 1, which means the nucleus sampling was disabled. However, this approach did not give us very good results. That is why we decided to adjust the top-k and top-p parameters, which significantly improved the performance, particularly with the nucleus sampling (top-p) strategy. We decided to set the top-k parameter to 20 and top-p to 0.8, a change that notably improved the quality of the summaries generated by our models.

Finally, for all models there were limitations in the maximal number of tokens that the model is allowed to generate, that was no more than 256 tokens.

2.4 Evaluation Metrics

In the introductory chapter, we had already discussed the types of NLG evaluation metrics. In this section, we provide a more detailed explanation of the metrics we used to evaluate our generated summaries. The first three metrics described here compare the generated text to the reference summary. The final one, SummaC, is a reference-free metric.

2.4.1 BERTScore

BERTScore [69] serves as an automated metric for evaluating text generation. It compares generated text to ground truth references. The authors of BERTScore claimed that it correlates better with human judgments compared to the majority of the existing metrics.

BERTScore computes a similarity score between each token in the generated sentence and each token in the reference sentence. Instead of exact matches, this method computes token similarity using contextual embeddings. Contextual embeddings can create unique vector representations of the same word in different sentences. These representations vary depending on the surrounding words, which define the word's context.

BERTScore Computation

Having a reference sentence and a candidate sentence, contextual embeddings are used to represent the tokens, after which cosine similarity is calculated to match the representations. A more detailed explanation of how the score is computed follows.

To begin with, the sentences are tokenized using the tokenizer provided with each model. This results in a tokenized generated sentence $x = \langle x_1, \ldots, x_k \rangle$ and a tokenized ground truth sentence $\hat{x} = \langle \hat{x}_1, \ldots, \hat{x}_n \rangle$. The embedding model is then used to generate a sequence of vectors $\langle \mathbf{x}_1, \ldots, \mathbf{x}_k \rangle$ and $\langle \widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n \rangle$, corresponding to the generated and ground truth tokenized sentences, respectively. These vector representations enable a soft measure of similarity, rather than relying on exact-string or heuristic matching.

The cosine similarity between a reference token x_i and a candidate token $\hat{x}_j$ is computed using the following equation:

$$\text{cosine\_similarity}(x_i, \hat{x}_j) = \frac{\mathbf{x}_i^T \widehat{\mathbf{x}}_j}{\|\mathbf{x}_i\| \|\widehat{\mathbf{x}}_j\|} \tag{2.2}$$

Because BERTScore is using pre-normalized vectors, the calculation is simplified to computing only the inner product $\mathbf{x}_i^T \widehat{\mathbf{x}}_j$. Even though this measure considers the tokens independently, the contextual embeddings capture information from the entire sentence. For computing the final score, greedy matching is applied to maximize the similarity score, where each token is paired with the most similar token from the other sentence. BERTScore recall is calculated by matching each token in x to a token in $\hat{x}$. Here is the formula:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i \cdot \hat{x}_j^\top \tag{2.3}$$

Each token in $\widehat{x}$ is matched with a token in x to compute precision. The following formula represents the BERTScore precision calculation:

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i \cdot \hat{x}_j^\top \tag{2.4}$$

Precision and recall are then combined to calculate the F1 score in the following way:

$$F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{2.5}$$

BERTScore versus Human Judgements

The authors of BERTScore [69] used human evaluations from the MSR Abstractive Text Compression Dataset [95] to show BERTScore's usefulness in abstractive text compression assessments. The dataset includes three types of human ratings: **meaning**, which refers to how well a compressed text preserves its original meaning; **grammar**, which evaluates the compressed text's grammatical correctness; and **combined**, which average the meaning and grammar scores. In the paper [69] was presented the Pearson correlation between BERTScore and the three different categories of human scores. R_{BERT} demonstrated the strongest link with human meaning judgments, whereas P_{BERT} had a significant correlation with human grammar rating. F_{BERT} provided a balanced review of the two aspects. It is important to have these things in mind when interpreting the results obtained by using BERTScore.

Model Selection

To evaluate English text generation, the authors of BERTScore recommended using the 24-layer RoBERTa-large model for BERTScore computation[69]. Consequently, we adopted this model for our evaluations to ensure the robustness and accuracy of our assessments.

2.4.2 Semantic Textual Similarity with Sentence Transformers

Sentence Transformers (also known as SBERT) is a widely used Python module for accessing, using, and training state-of-the-art models for text and image embeddings. It was introduced by the authors of the following paper [96], who were motivated by the observation that BERT, in its original form, maps sentences to a vector space that is not well-suited for common similarity measures such as cosine similarity. To address this limitation, they proposed Sentence-BERT (SBERT), which fine-tunes BERT using a siamese/triplet network architecture. An additional key advantage of SBERT is its high computational efficiency.

We used this Python module to calculate the Semantic Textual Similarity (STS) for our pairs of generated and ground truth summaries. Cosine similarity was applied for the similarity calculation. The model that we used for our evaluation was "all-MiniLM-L6-v2" that is available in the HuggingFace models hub.

2.4.3 Sentence Distance-Based Evaluation Metrics

Other metrics for automatic evaluation of AI-generated text in a continuous space using word and sentence embeddings are Word Mover's Similarity (WMS), Sentence Mover's Similarity (SMS), and Sentence and Word Mover's Similarity (S+WMS) [70]. Their authors found that these sentence distance-based metrics had a significantly higher correlation with human judgments than ROUGE [65]. That is why we are going to use them to evaluate our generated summaries, and in the next paragraphs we will explain them in more details.

Word Mover's Similarity

WMS evaluates document similarity by minimizing the total distance required to move words between two documents, combining the benefits of bag-of-words (BOW) and word embedding-based metrics [70]. Follows an explanation on how it is calculated, starting from the Word Mover's Distance (WMD) [97] calculation.

- **Document Representation:** The two documents that are being compared are represented by calculating the relative frequencies of words each of them contains. Here is the formula for a given document A:

$$d_{A,i} = \frac{\text{count}(i)}{|A|} \tag{2.6}$$

  In the formula $|A|$ represents the total word count.

- **Word Embeddings and Distance Calculation:** Each word i is represented by a vector $\mathbf{v}_i \in \mathbb{R}^m$ (embedding of length m). The Euclidean distance between words i and j is defined as:

$$\Delta(i,j) = \|\mathbf{v}_i - \mathbf{v}_j\|_2 \tag{2.7}$$

- **WMD Calculation:** The WMD between documents A and B is computed by solving the following linear program:

$$\text{WMD}(A,B) = \min_{T \geq 0} \sum_{i=1}^{V} \sum_{j=1}^{V} T_{i,j} \Delta(i,j) \tag{2.8}$$

  subject to

$$\sum_{j=1}^{V} T_{i,j} = d_{A,i}, \quad \forall i \tag{2.9}$$

$$\sum_{i=1}^{V} T_{i,j} = d_{B,j}, \quad \forall j \tag{2.10}$$

  In the formula V is the size of the vocabulary. T is a non-negative matrix, and $T_{i,j}$ represents how much of word i's tokens in the document A are assigned to tokens of word j in B. The constraints guarantee that the flow of a given word does not surpass its weight.

- **Transformation to WMS:** To convert WMD to a similarity score, the authors of the paper [70] apply an exponential transformation:

$$\text{WMS}(A, B) = \exp(-\text{WMD}(A, B)) \tag{2.11}$$

Sentence Mover's Similarity

SMS [70] operates with bag of sentence embeddings rather than word embeddings. While WMS calculates the distance between word embeddings weighted by their frequency in a document, SMS uses sentence embeddings, which are derived by averaging word embeddings. In SMS, each sentence embedding is weighted by the number of words it contains rather than by word frequency, and the goal is to compute the cumulative distance of moving a document's sentences to match another document.

Sentence and Word Mover's Similarity

S+WMS [70] combines WMS and SMS, representing each document as a collection of word and sentence embeddings. Word embeddings are weighted by frequency, whereas sentence embeddings are weighted by length. The method solves the same linear optimization problem as WMS and SMS but instead calculates the cumulative distance of moving words and sentences to align two documents.

2.4.4 SummaC

SummaC (Summary Consistency) [74] is a benchmark for evaluating the factual consistency of summaries by detecting inconsistencies between the summary and its input document. Natural Language Inference (NLI), also known as textual entailment, is a task that requires a hypothesis sentence to be classified as either entailed by, neutral, or contradictory to a premise sentence $\text{SummaC}_{\text{Conv}}$ is an NLI-based model for identifying summary inconsistencies. It is based on the authors' key insight that NLI models perform best with sentence-level input [74]. The $\text{SummaC}_{\text{Conv}}$ model architecture will be briefly described in the following paragraph.

$\text{SummaC}_{\text{Conv}}$ Model

The first step in the process is to apply a pretrained NLI model to generate an NLI Pair Matrix for a (document, summary) pair. This matrix is generated by dividing the document and summary into sentence blocks. The document is separated into M blocks, each serving as a premise labeled from D_1 to D_M. The summary is divided into N blocks, each acting as a hypothesis labeled from S_1 to S_N. Every combination of D_i and S_j is processed by the NLI model, which produces a probability distribution over the three NLI categories: entailment, contradiction, and neutral (E_{ij}, C_{ij}, and N_{ij}). The pair matrix has dimensions $M \times N$ and it contains the entailment scores E_{ij}.

The $\text{SummaC}_{\text{Conv}}$ algorithm first converts each column of the NLI Pair Matrix into a fixed-size histogram that represents the score distribution for the corresponding summary sentence. The NLI scores are then divided into H equally sized bins. The resulting binned matrix is processed through a one-dimensional convolution layer with

a kernel size of H, which scans the summary histograms sequentially and combines them into a scalar value for the corresponding summary. The final summary-level score is calculated by averaging the scores from all summary sentences. To update the convolution layer weights, the $\text{SummaC}_{\text{Conv}}$ model is trained with FactCC synthetic training data [98]. For more information on the training procedure, refer to their paper [74].

3

Results and Discussion

3.1 Dataset Complexity Impact on Summary Quality

In this section, we examine the effects of increasing dataset complexity on the quality of generated summaries. As outlined in the methods chapter, we created three datasets to represent different levels of input information for training: the "GOTerms" dataset, which includes inputs based only on Gene Ontology terms; the "Prop-GOTerms" dataset, where GO terms were further propagated through their hierarchical relationships; and the "GOTerms + SwissProt" dataset, which integrates additional protein functional descriptions. Using these datasets, we evaluate how each level of complexity influences the model's ability to generate accurate and informative summaries.

To achieve this, we fine-tuned the pre-trained T5 model with the three datasets. Each model was fine-tuned over three epochs, with the ROUGE-based validation loss recorded across epochs to monitor learning progress. Figure 3.1 shows the loss trajectory for each model during the fine-tuning process. Notably, loss trends vary significantly between models trained with different datasets. Specifically, the T5 model fine-tuned with the "GOTerms + SwissProt" dataset shows a more significant decrease in loss than the other two models, indicating that the most complex input probably is the most comprehensive and helpful for this task.

After fine-tuning, we compared the generated summaries to ground truth summaries using the previously mentioned reference-based evaluation metrics: BERTScore, STS with the MiniLM model, and Sentence Distance-Based Metrics (SMS, WMS, and S+WMS). As we explained in the methods chapter, these methods output a single score for each reference and generated summary pair. Our test set includes 2405 samples. For example, suppose that for a given sample, Model 1 generates a summary with a score of 0.85 according to some metric, Model 2 has a score of 0.83, and Model 3 has a score of 0.81. In this case, Model 1 would be considered the best at generating a summary for that sample, resulting in a "win" for that model. This process is repeated for all samples in the test set, with the number of "wins" for each model recorded. The results are presented as percentages on Figure 3.2,

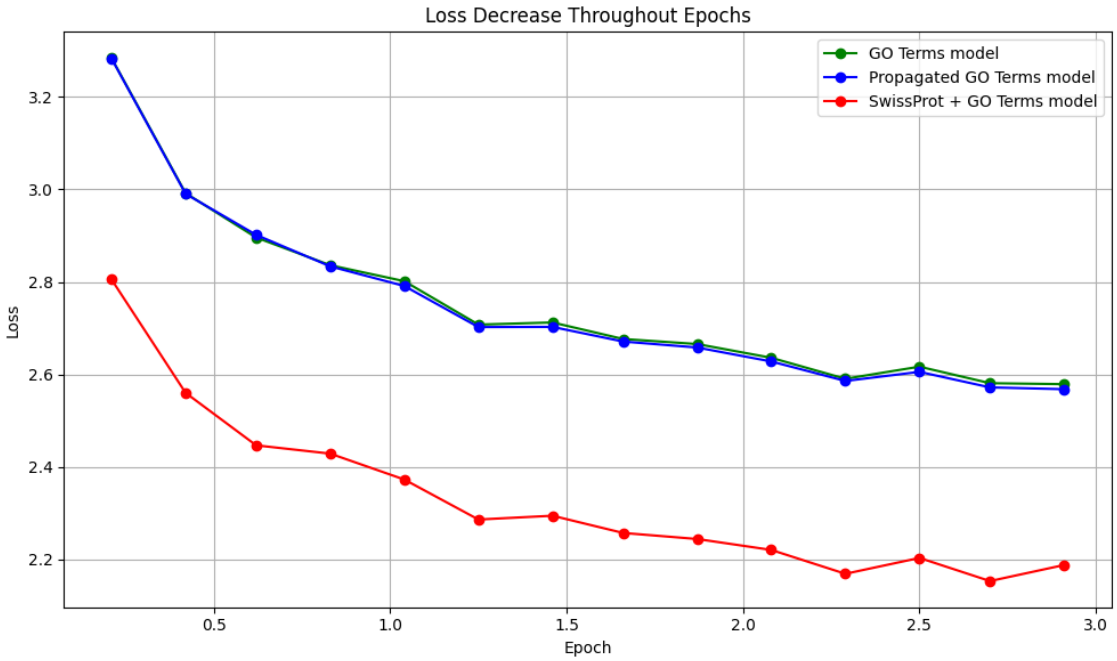

Figure 3.1: Loss decrease over the epochs while fine-tuning T5 models with the "GOTerms" dataset (green), "PropGOTerms" dataset (blue), and "GOTerms + SwissProt" dataset (red).

indicating the proportion of samples for which each model produced a summary that was closest to the ground truth. The plots show that the T5 model, fine-tuned using the most complex "GOTerms + SwissProt" dataset, has exhibited the best performance across all metrics, generating the most similar summaries to the references in more than 60% of the samples from the test set. Based on the BERTScore recall metric, the "GOTerms + SwissProt" model outperforms other models in producing summaries for approximately 70% of test set samples.

To better understand each model's performance, we plotted the distribution of BERTScore F1 scores for summaries generated by the three fine-tuned T5 models. It is shown on Figure 3.3. The histograms allow us to see the distribution and central tendency of F1 scores across all test samples for each model. The plot shows that the three distributions overlap considerably, indicating that all models perform similarly on a portion of the test set. However, the tail of the model fine-tuned with the "GOTerms + SwissProt" dataset extends further to the right, indicating that this model generates higher-scoring summaries for certain samples. This distribution analysis supports the conclusion that dataset complexity positively impacts model performance. Table 3.1 shows the average BERTScore precision, recall, and F1 scores across the test set for the three models. While these average scores give a general sense of each model's performance, they should be interpreted together with the other results, which provide a more detailed view of score distributions and model differences.

On Figure 3.4, there are scatter plots showing pairwise comparisons of MiniLM STS scores for the summaries generated by each model on the test set. Additionally, we calculated Pearson and Spearman correlation coefficients. The results demonstrate strong correlations between the fine-tuned T5 models trained on the "GOTerms" and "PropGOTerms" datasets, with Spearman and Pearson correlation coefficients of 0.842 and 0.845, respectively,

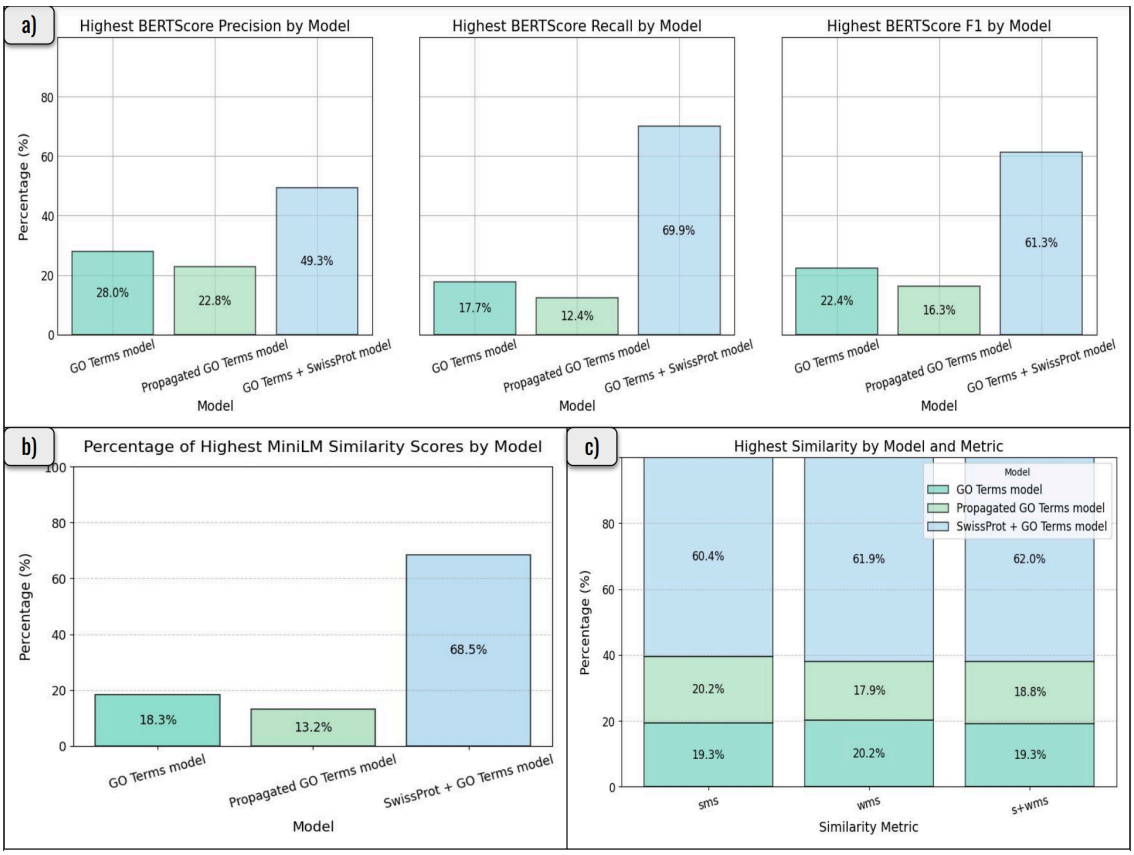

Figure 3.2: The plots show the proportion of test set entries where each model produced a summary that is closer to the ground truth compared to the summaries from the other models, based on the highest similarity score for each metric. **a)** BERTScore (Precision, Recall, F1) metric; **b)** STS with MiniLM model; **c)** SMS, WMS, and S+WMS metrics.

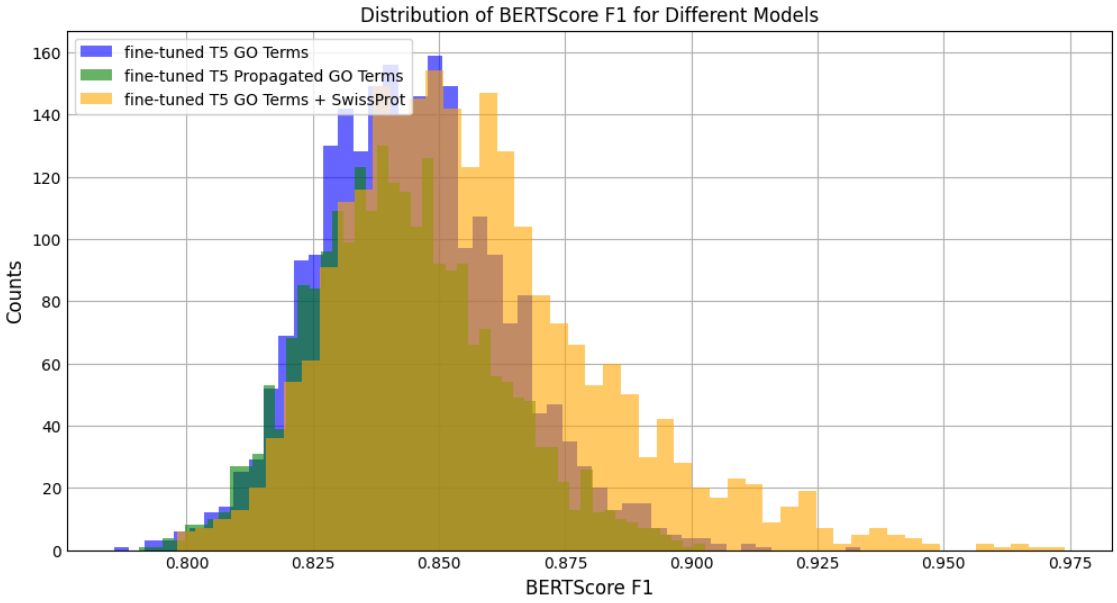

Figure 3.3: Distribution of BERTScore F1 scores for summaries generated by fine-tuned T5 models using different input datasets.

Model	Average Precision	Average Recall	Average F1
Fine-tuned T5 model with the "GOTerms" dataset	0.857	0.833	0.845
Fine-tuned T5 model with the "PropGOTerms" dataset	0.855	0.831	0.843
Fine-tuned T5 model with the "GOTerms + SwissProt" dataset	**0.866**	**0.848**	**0.857**

Table 3.1: Average BERTScore precision, recall, and F1 scores for each fine-tuned T5 model across all test set samples.

both statistically significant. This suggests that propagating the GO terms had no significant effect on the quality of the generated summaries.

In contrast, the correlations between these models and the model fine-tuned with the "GOTerms + SwissProt" dataset are moderate, with Pearson and Spearman coefficients of approximately 0.4. The summaries generated by the "GOTerms + SwissProt" model had significantly higher scores than the other models' summaries, supporting our previous conclusions that including protein descriptions and organism data to the input improves the model's ability to generate better summaries. Therefore, for all subsequent experiments described in the following sections, we used the "GOTerms + SwissProt" dataset.

3.2 Pre-trained vs Fine-Tuned T5 Model Comparison

Comparing the performance of pre-trained and fine-tuned T5 models for summarization provides insights into the effectiveness of task-specific training. We've already mentioned that the pre-trained T5 model is trained on a large corpus and can produce general summaries, but it may lack domain-specific accuracy. The process of fine-tuning allows the model to adapt to the specific patterns and complexities of the dataset and produce more relevant

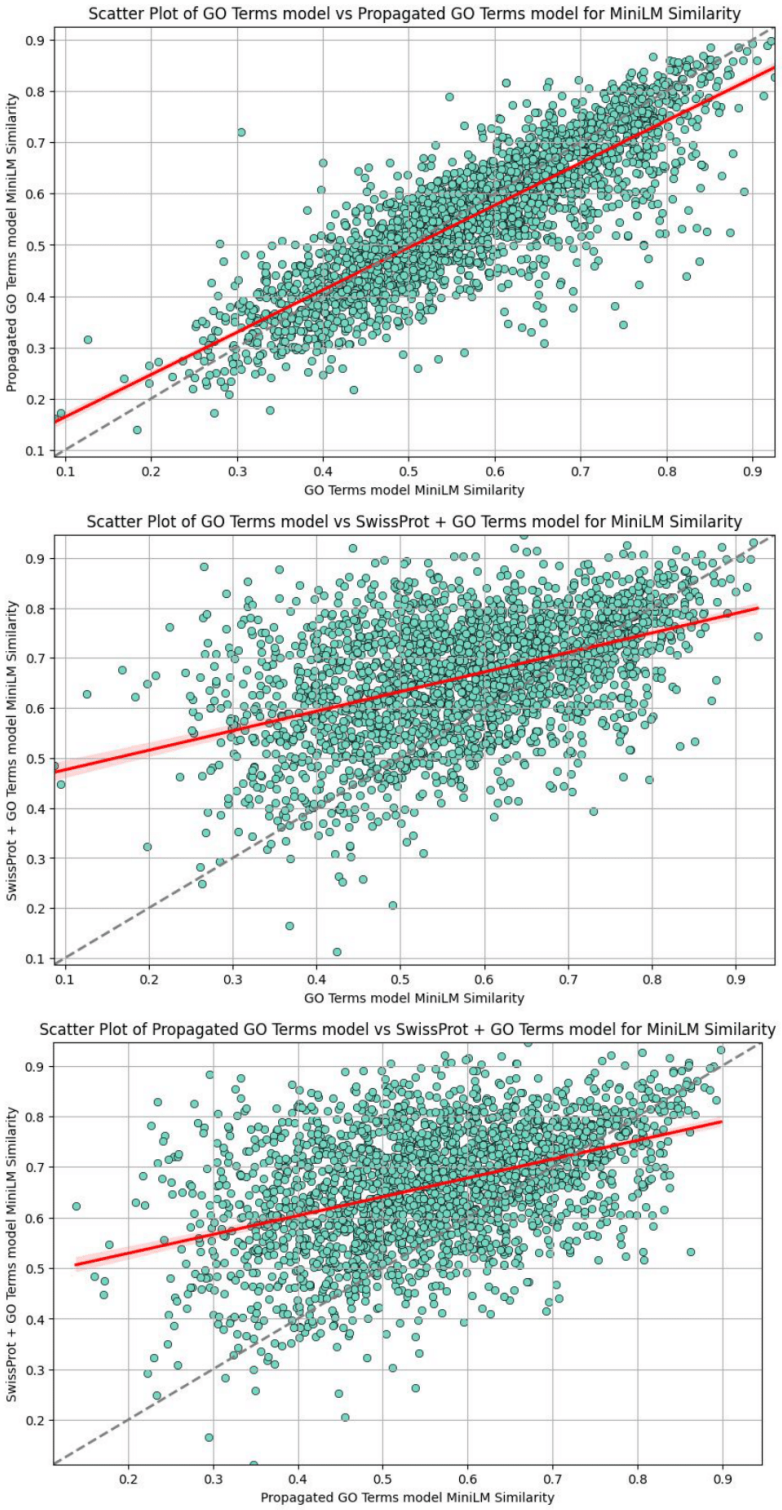

Figure 3.4: Scatter plots showing pairwise correlations of MiniLM STS scores between the three T5 models fine-tuned with different input datasets. Each plot contains a red regression line.

summaries.

In this study, we evaluated the impact of fine-tuning by generating summaries for the "GOTerms + SwissProt" test set with both pre-trained and fine-tuned T5 models and comparing their outputs. Evaluation scores were calculated using the same metrics as described in the previous section.

Figure 3.5 shows the proportion of test set samples where each model's summary achieved the highest similarity to the ground truth based on evaluation metrics. The fine-tuned T5 model consistently outperforms the pre-trained model across all metrics. For BERTScore (Precision, Recall, and F1), the fine-tuned model demonstrates particularly strong performance in recall, achieving higher recall scores in 96.9% of cases compared to the pre-trained model. Similarly, with the MiniLM STS metric, the fine-tuned model generates better summaries in 85.5% of cases. Sentence distance-based metrics further confirm this trend. These findings highlight the importance of fine-tuning in increasing the ability of the model to provide accurate and domain-specific summaries.

To further assess model performance, we plotted the BERTScore F1 score distributions for summaries generated by the pre-trained and fine-tuned T5 models, as shown in Figure 3.6. The distribution for the fine-tuned model is notably shifted to the right, indicating consistently higher F1 scores. This observation confirms that fine-tuning enhances summary quality, as the model performs better across a broader range of test data. In addition, we also provided the average BERTScore precision, recall, and F1 scores for the two models in Table 3.2 to further support our findings.

Model	Average Precision	Average Recall	Average F1 Score
Pre-trained vanilla T5 model	0.851	0.815	0.832
Fine-tuned T5 model	**0.866**	**0.848**	**0.857**

Table 3.2: Average BERTScore precision, recall, and F1 scores for summaries generated for the "GOTerms + SwissProt" test set by the pre-trained vanilla T5 model and the fine-tuned T5 model.

The scatter plot in Figure 3.7 shows that the majority of points fall in the upper triangle, further confirming the improved quality of summaries generated by the fine-tuned model. However, there are still some points where the pre-trained model generated summaries with higher MiniLM STS scores than the fine-tuned model. To investigate these cases, we created Table 3.3, which includes one example where the pre-trained model outperforms the fine-tuned model and vice versa. For the entry with InterPro ID "IPR001136," the fine-tuned model summary captures the key aspects described in the original, while the pre-trained model summary is irrelevant. In contrast, for the "IPR037143" entry, both summaries are of poor quality. The pre-trained model's summary includes relevant terms such as "magnesium ion binding," which explains its higher score, but the overall summary lacks meaning. The fine-tuned model, on the other hand, focuses on unimportant details, resulting in a summary that does not accurately reflect the original.

These findings emphasize the importance of fine-tuning in the task of generating protein signature descriptions. However, they also show that many cases still have room for improvement, implying that careful consideration of dataset characteristics and further refinement is critical for achieving the best results.

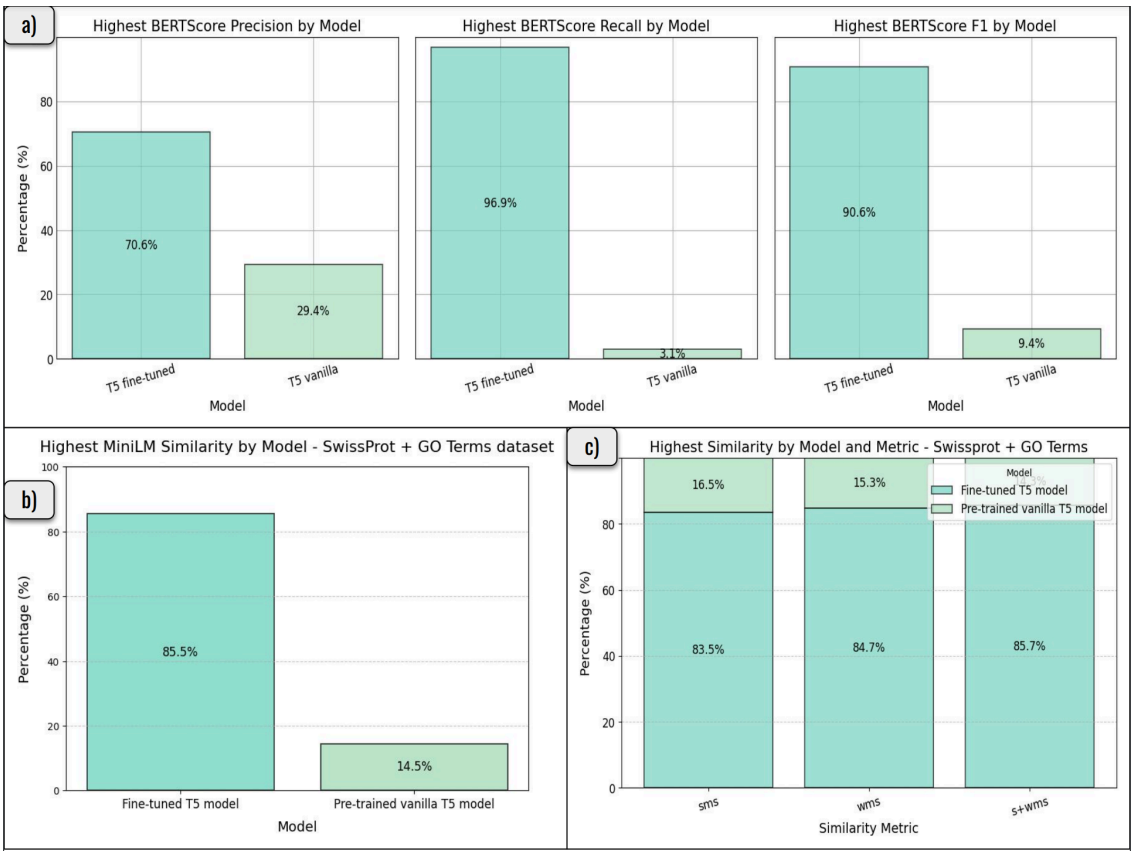

Figure 3.5: The plots show the proportion of test set entries where the pre-trained vanilla T5 model or the fine-tuned T5 model produced a summary that is closer to the ground truth, based on the highest similarity score for each metric. **a)** BERTScore (Precision, Recall, F1) metric; **b)** STS with MiniLM model; **c)** SMS, WMS, and S+WMS metrics.

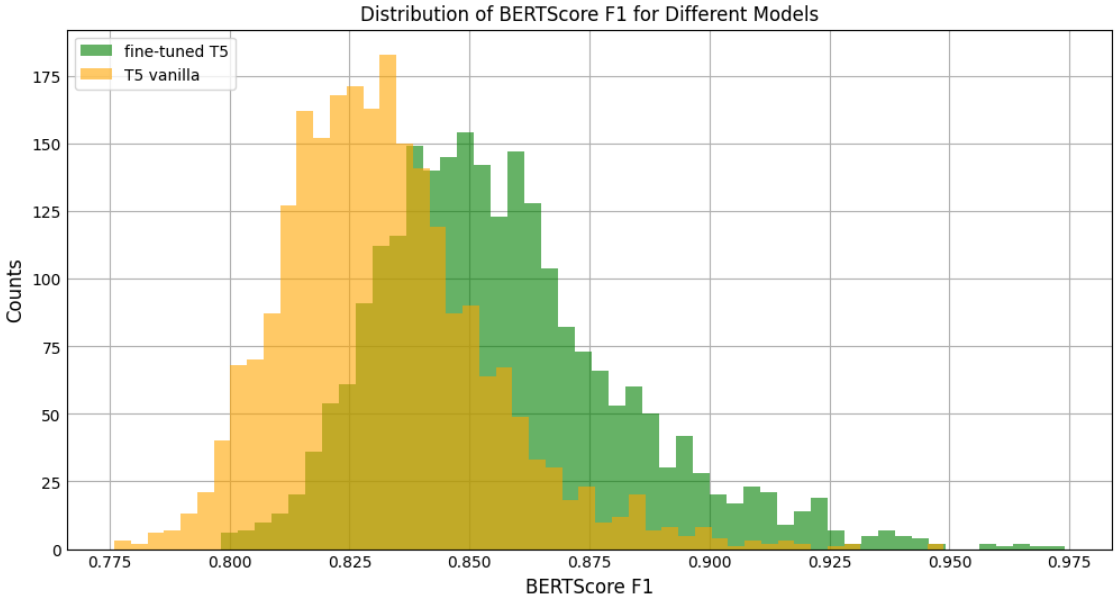

Figure 3.6: Distribution of BERTScore F1 scores for summaries generated by the pre-trained T5 model and the fine-tuned T5 model with the "GOTerms + SwissProt" dataset.

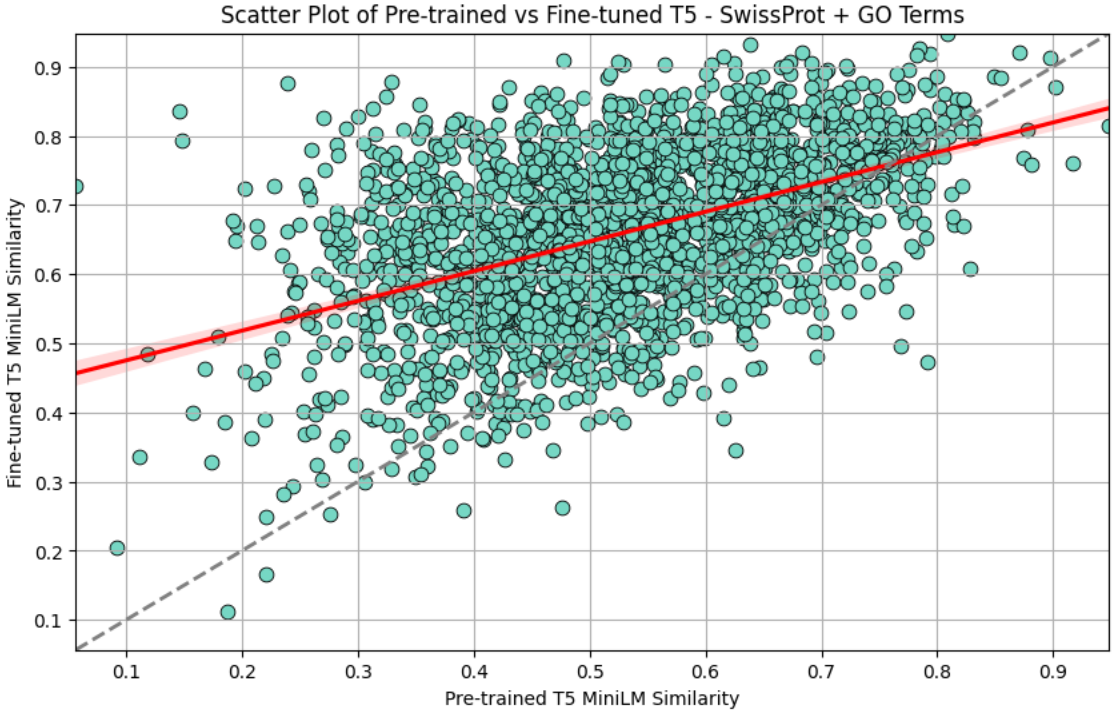

Figure 3.7: Scatter plot showing correlation of MiniLM STS scores between the pre-trained and fine-tuned T5 model with the "GOTerms + SwissProt" dataset.

InterPro ID	Original Summary	Fine-tuned T5 Summary	Score (Fine-tuned)	Vanilla T5 Summary	Score (Vanilla)
IPR001136	This entry represents a protein family specific to the genus Plasmodium. The merozoite surface antigen 2 may play a role in the merozoite attachment to the erythrocyte. This protein was proposed to be a candidate for a protective vaccine against malaria.	This entry represents a group of merozoite-associated proteins from Plasmodium falciparum. They may play a role in the erythrocyte attachment.	**0.829**	cell adhesion: the attachment of a cell to another cell or to an underlying substrate. it corresponds to Swiss-Prot proteins, with their functions and associated organisms listed below.	0.3
IPR037143	The 4'-phosphopantetheinyl transferase domain superfamily of proteins transfer the 4'-phosphopantetheine moiety from coenzyme A to the invariant serine of pp-binding. This post-translational modification renders holo-ACP capable of acyl group activation via thioesterification of the cysteamine thiol of 4'-PP. This superfamily consists of two subtypes: The ACPS type such as ACPS_ECOLI and the Sfp type such as SFP_BACSU. The structure of the Sfp type is known, which shows the active site accommodates a magnesium ion. The most highly conserved regions of the alignment are involved in binding the magnesium ion.	Fatty acid synthase catalyses the formation of long-chain fatty acids from coenzyme A, malonyl-CoA and NADPH. The alpha subunit contains domains for: acyl carrier protein, 3-oxoacyl-reductases, and 3'-octaacyl synthetases. This entry represents the N-terminal domain of the alpha-subunit of the fatty acid holo-synthetic enzymes. This domain is found in fungi, bacteria, plants and bacteria.	0.474	the gene ontology terms for the given Homologous superfamily are magnesium ion binding. the transfer of the 4'-phosphopantetheine (Ppant) co-factor from coenzyme A to the hydroxyl side chain of the serine residue of the acyl-carrier protein (ACP) corresponds to Swiss-Prot proteins, with their functions and associated organisms listed.	**0.791**

Table 3.3: Examples of summaries generated by the fine-tuned and vanilla T5 models, along with their respective MiniLM STS scores.

3.2.1 Input Features and Summary Quality

We explored how certain input features from the dataset might influence the evaluation scores of summaries generated by the fine-tuned T5 model. Specifically, we analyzed the number of GO terms, number of protein descriptions, and input length (number of tokens) in relation to the BERTScore F1 scores for the test set summaries.

To quantify these relationships, we calculated Pearson and Spearman correlations between each input feature and the F1 scores. The analysis revealed no correlation between the number of GO terms and the F1 scores. However, input length showed a weak negative Pearson correlation of -0.297 and a Spearman correlation of -0.3, both statistically significant. Similarly, the number of protein descriptions also displayed weak negative correlations, with Pearson and Spearman values of -0.243 and -0.262, respectively. The plots illustrating these correlations are shown in Figure 3.8. These findings suggest that inputs with more tokens or protein descriptions tend to produce summaries with slightly lower BERTScore F1 scores, though the impact is minimal.

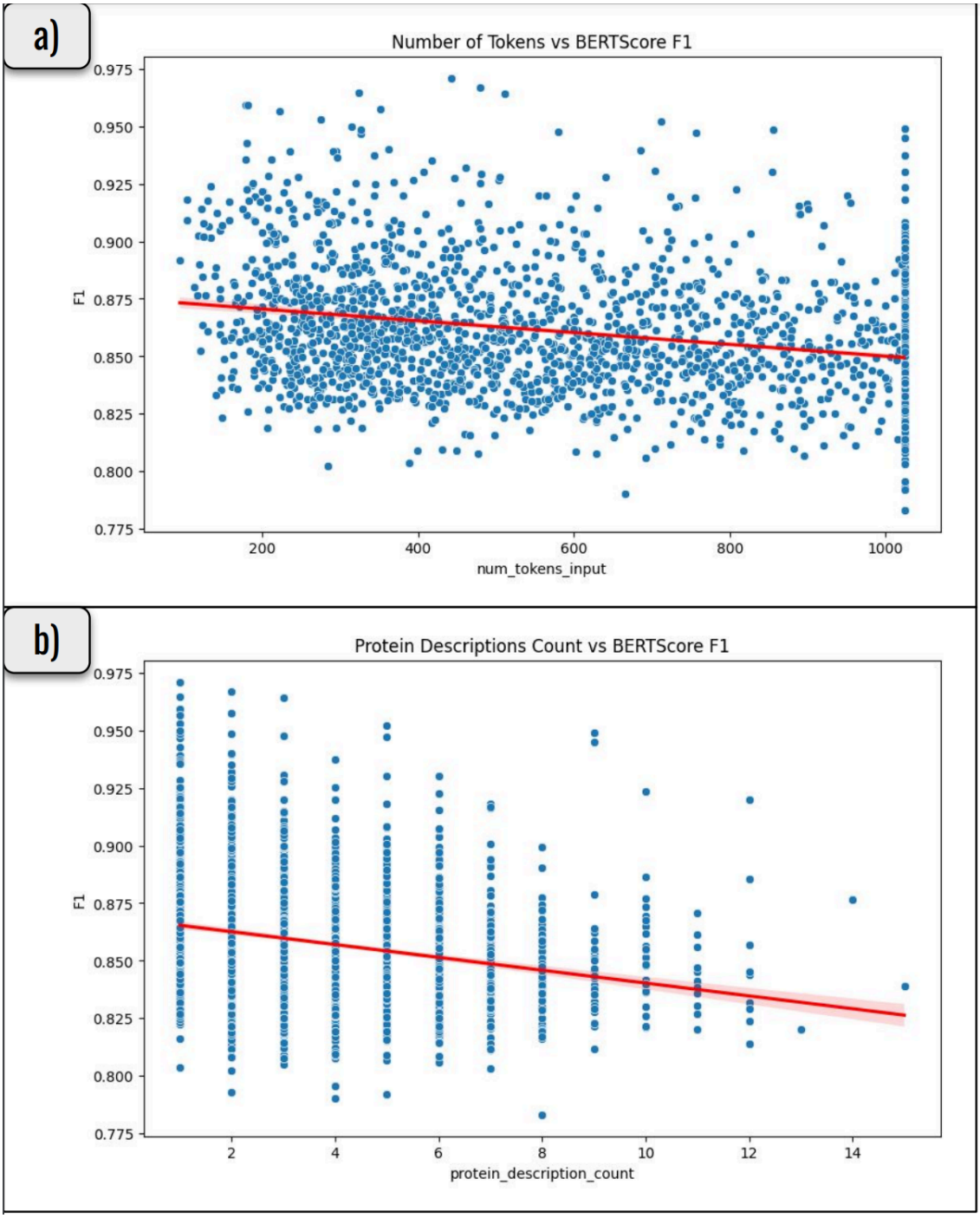

Figure 3.8: Relationship between input features and BERTScore F1. (a) Scatter plot illustrating the relationship between the number of tokens in the input and BERTScore F1. (b) Scatter plot showing the relationship between the number of protein descriptions and BERTScore F1.

3.3 Comparing Fine-Tuned T5, GPT-2, and BioGPT

In addition to fine-tuning the T5 model, we also fine-tuned two other pre-trained models, GPT-2 and BioGPT, to compare the quality of the generated summaries. Given that the BioGPT model has been pre-trained on biological data, we were particularly interested in its performance. In all experiments, the models were fine-tuned using the "GOTerms + SwissProt" dataset. Each model was initially fine-tuned for up to 10 epochs. However, the BioGPT model began to overfit after the sixth epoch, as evidenced by an increase in validation loss, prompting us to stop training at that point. In contrast, the other models showed no signs of overfitting, with validation loss continuing to decrease until the tenth epoch.

In the introductory chapter, we discussed different types of decoding strategies, noting that beam search is typically preferred for directed tasks, while sampling strategies are more suitable for open-ended tasks. Given that our task likely falls between open-ended and directed generation, we decided to evaluate both decoding strategies with our fine-tuned models. We will first report the results obtained using the beam-search decoding strategy, followed by those acquired with the sampling strategy. Finally, in the section "Different Decoding Strategies Results," we will compare the performance of the best-performing models for each strategy.

3.3.1 Beam-Search Decoding Results

We generated summaries for all samples in the test set using the beam-search decoding strategy with our fine-tuned T5, GPT-2, and BioGPT models. As it was already explained in the methods chapter, the text generation process was configured with a beam size of 4, a length penalty of 2, a no-repeat n-gram size of 3, and early stopping enabled.

Figure 3.9 presents bar plots illustrating the proportion of test set entries where each of the fine-tuned models produced summaries that were closest to the ground truth compared to the other two models. These comparisons are based on the highest similarity scores obtained for each evaluation metric. Notably, the fine-tuned T5 model generated summaries most similar to the original ones in over 40% of the test samples across all metrics, making it the best-performing model under the beam-search decoding strategy.

The distribution of BERTScore F1 scores for the models is shown in Figure 3.10, where significant overlap is evident in the histograms. This indicates that for many test samples, all three models produced summaries of comparable quality. The average BERTScore precision, recall, and F1 scores for the T5 model surpass those of GPT-2 and BioGPT, as shown in Table 3.4. This further confirms that when using the beam-search decoding strategy, this model produces slightly better summaries compared to the other two models.

Model	Average Precision	Average Recall	Average F1 Score
Fine-tuned T5 model	**0.866**	**0.851**	**0.858**
Fine-tuned GPT-2 model	0.859	0.841	0.850
Fine-tuned BioGPT model	0.857	0.848	0.852

Table 3.4: Average BERTScore precision, recall, and F1 scores for summaries generated with beam-search decoding strategy by the fine-tuned T5, GPT-2, and BioGPT models.

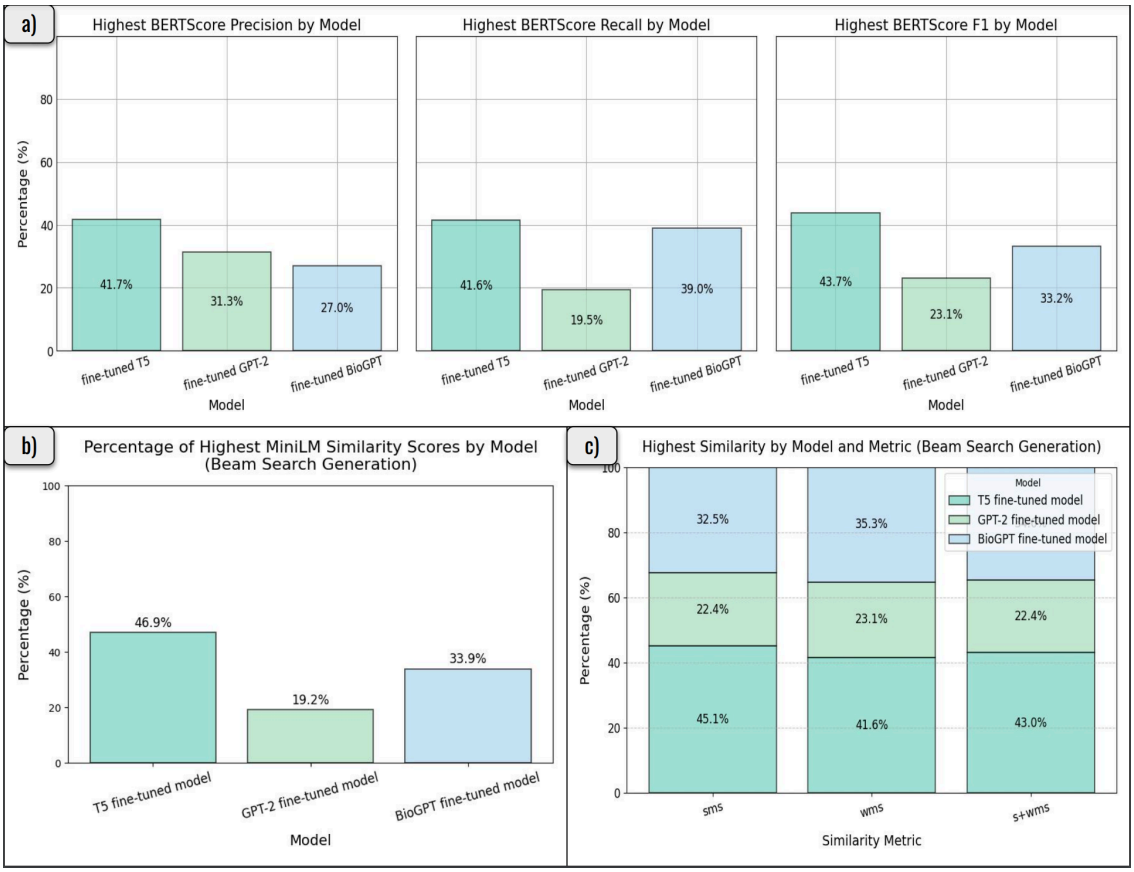

Figure 3.9: The plots show the proportion of test set entries where the fine-tuned T5, GPT-2, and BioGPT models generated a summary that is closer to the ground truth compared to the summaries from the other models, based on the highest similarity score for each metric. Beam-search decoding strategy was used. **a)** BERTScore (Precision, Recall, F1) metric; **b)** STS with MiniLM model; **c)** SMS, WMS, and S+WMS metrics.

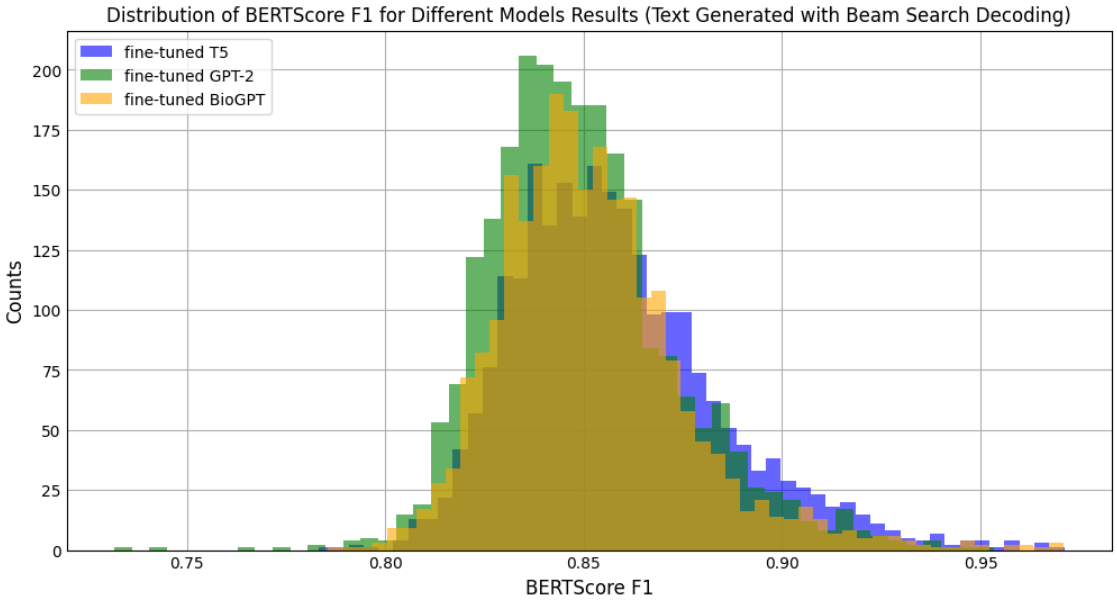

Figure 3.10: Distribution of BERTScore F1 scores for summaries generated with beam-search decoding strategy by the fine-tuned T5, GPT-2, and BioGPT models.

3.3.2 Sampling Results

In this experiment, instead of generating summaries with beam-search decoding, we used sampling as the decoding strategy for all of our models. As described in the methods chapter, the hyperparameters for text generation, top-k and top-p, were set to 20 and 0.8, respectively.

The plots on Figure 3.11 show a change in the results compared to those obtained with beam-search decoding. With sampling, the fine-tuned BioGPT model produces the most similar summaries to the ground truth summaries with regard to all evaluation metrics, with the exception of BERTScore precision, where the fine-tuned T5 model performs slightly better.

The distribution of BERTScore F1 scores for all three models is shown in Figure 3.12, and, as with beam-search, there is significant overlap in the histograms. The average BERTScore precision, recall, and F1 scores for summaries generated using nucleus sampling by our models are shown in Table 3.5. The fine-tuned BioGPT achieved the highest average in all metrics except for precision.

These results demonstrate that the decoding strategy significantly impacts the quality of generated text. By merely switching from beam-search to sampling, we observed substantial differences in the results. Previously, the fine-tuned T5 model produced the most similar summaries to the originals, whereas the fine-tuned BioGPT model now shows slightly better performance with the sampling strategy. Given the variation in model performance with different decoding strategies, the next section will compare the generated summaries by the fine-tuned T5 and BioGPT models using both decoding strategies. Specifically, we will compare the summaries generated with beam-search decoding by the fine-tuned T5 model against those generated with sampling by the fine-tuned BioGPT model.

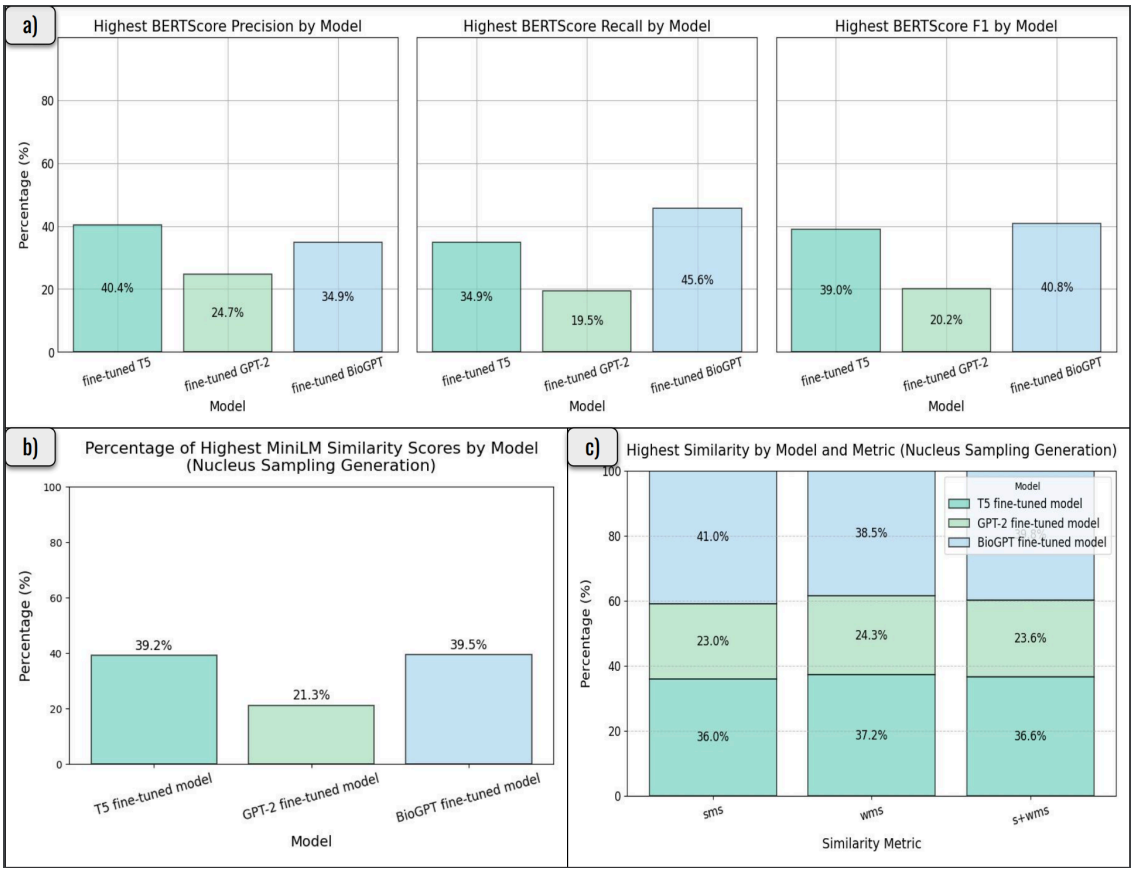

Figure 3.11: The plots show the proportion of test set entries where the fine-tuned T5, GPT-2, and BioGPT models generated a summary that is closer to the ground truth compared to the summaries from the other models, based on the highest similarity score for each metric. Sampling was used as a decoding strategy. **a)** BERTScore (Precision, Recall, F1) metric; **b)** STS with MiniLM model; **c)** SMS, WMS, and S+WMS metrics.

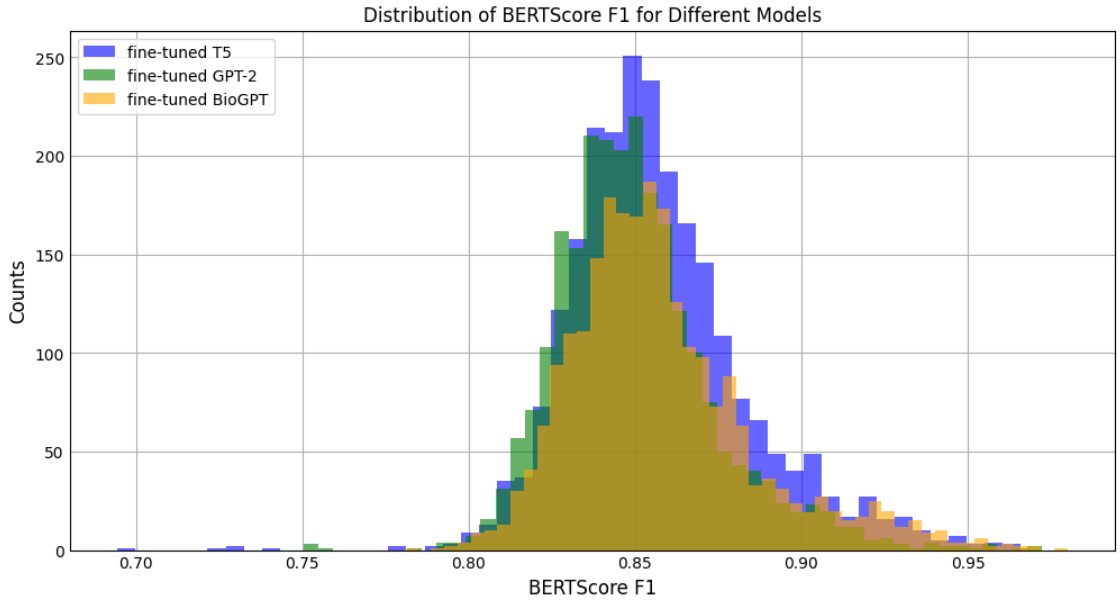

Figure 3.12: Distribution of BERTScore F1 scores for summaries generated with sampling as decoding strategy by the fine-tuned T5, GPT-2, and BioGPT models.

Model	Average Precision	Average Recall	Average F1 Score
Fine-tuned T5 model	**0.867**	0.848	0.857
Fine-tuned GPT-2 model	0.860	0.841	0.850
Fine-tuned BioGPT model	0.866	**0.851**	**0.858**

Table 3.5: Average BERTScore precision, recall, and F1 scores for summaries generated with sampling by the fine-tuned T5, GPT-2, and BioGPT models.

3.3.3 Different Decoding Strategies Results

In this section, we present the results of comparing summaries generated using beam-search decoding with the fine-tuned T5 model to those generated using nucleus sampling with the fine-tuned BioGPT model. As shown in Figure 3.13, the fine-tuned T5 model outperforms the fine-tuned BioGPT model in most evaluation metrics, achieving higher scores in BERTScore precision, BERTScore F1 score, MiniLM similarity, and all sentence distance-based metrics. The only exception is BERTScore recall, where the fine-tuned BioGPT model performs slightly better.

These results suggest that, with the chosen decoding strategies and hyperparameters, the fine-tuned T5 model has a slight advantage in generating accurate summaries. However, the overall similarity in performance suggests that the quality of the generated summaries is comparable between the two models. While the fine-tuned T5 model appears to be slightly more reliable, the differences are not significant enough to conclusively favor one model over the other. For the following experiments, which include generating descriptions for the "IDRs" dataset, we decided to use the fine-tuned T5 model. This decision was made to simplify the process and avoid

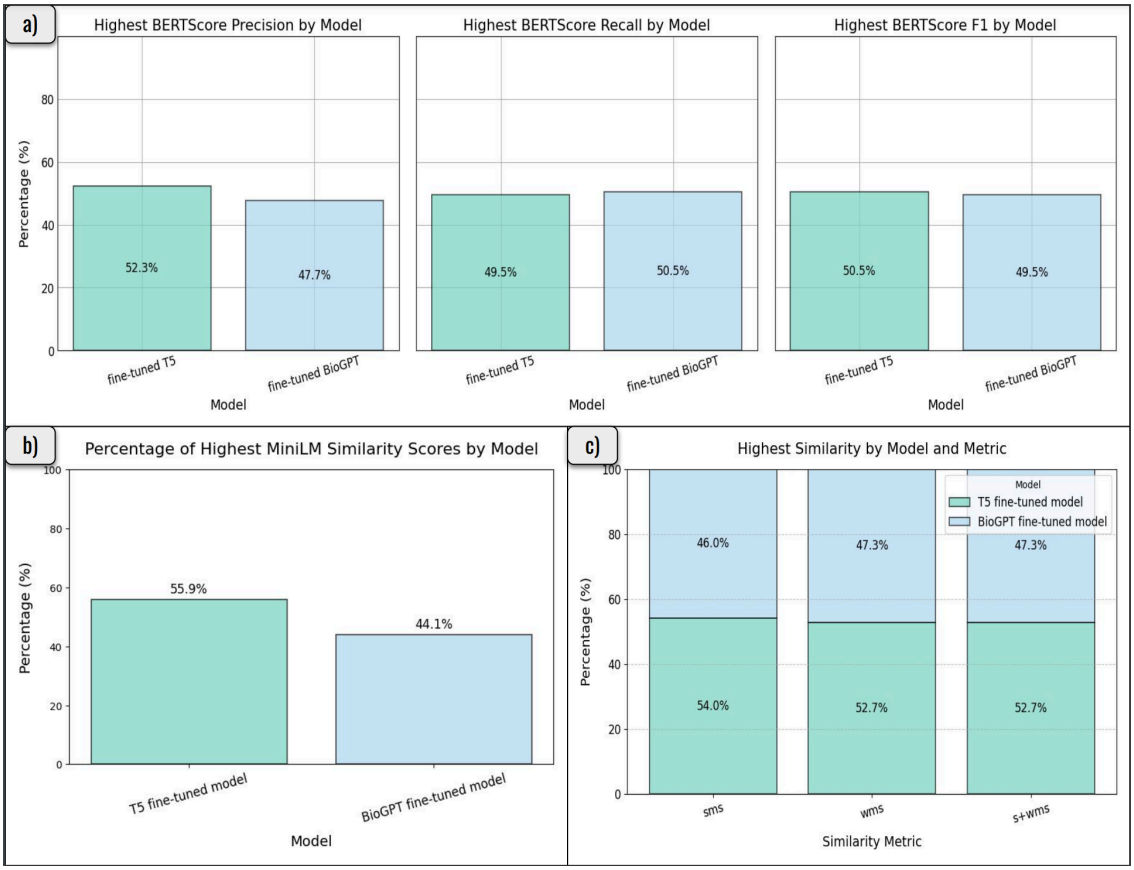

Figure 3.13: The plots show the proportion of test set entries where the fine-tuned T5 model (beam-search decoding) or the fine-tuned BioGPT model (nucleus sampling) produced a summary that is closer to the ground truth, based on the highest similarity score for each metric. **a)** BERTScore (Precision, Recall, F1) metric; **b)** STS with MiniLM model; **c)** SMS, WMS, and S+WMS metrics.

the additional effort of using both models, given the minimal performance differences.

3.4 Evaluation Metrics Scores Correlation

As mentioned in previous chapters, the "IDRs" dataset, for which we will build summaries using our fine-tuned T5 model, lacks ground truth summaries. This prevents us from using reference-based evaluation metrics, which we have used in all past evaluations. Instead, we intend to employ a reference-free evaluation metric, specifically the SummaC$_{\text{Conv}}$ model described in the methods chapter. To check for factual inconsistencies, this approach compares the output summary to its input document rather than a reference summary.

While we acknowledge that SummaC$_{\text{Conv}}$ evaluates summaries differently than reference-based metrics, we wanted to see if there is any correlation between the scores obtained using both types of metrics. To do this, we calculated scores using all evaluation metrics, including SummaC$_{\text{Conv}}$, for summaries generated by the fine-tuned

T5 model on our test set. Afterwards, we calculated Pearson and Spearman correlations for each pair of scores.

Pearson correlation measures the linear relationship between variables, whereas Spearman correlation measures monotonic relationships, in which variables tend to move in the same or opposite direction but not always at the same rate. In our analysis, we found weak linear correlations between some of the scores (for example, "SMS" and "WMS" had a Pearson correlation coefficient of 0.21), but the Spearman correlation was much stronger (0.76 between "SMS" and "WMS" scores). This is why we are showing the Spearman correlations for all metrics in Figure 3.14. The figure shows that reference-based metrics in general have moderate to strong positive correlations, ranging from 0.3 to 0.98. Notably, the "WMS" and "S+WMS" scores have a nearly perfect Spearman correlation (0.98), indicating redundancy—either metric could replace the other. However, $\text{SummaC}_{\text{Conv}}$ has a very weak negative correlation with all reference-based metrics. This suggests that $\text{SummaC}_{\text{Conv}}$ can be used in conjunction with reference-based metrics, not as a replacement for them.

In conclusion, $\text{SummaC}_{\text{Conv}}$ is useful for evaluating summaries in the absence of ground truth references. However, its low correlation with reference-based metrics indicates that it captures a different aspect of summary quality and should ideally be used in combination with other metrics for a more comprehensive assessment.

3.5 Generating Descriptions for IDRs

As stated in the introduction, one of the primary goals of this study was to generate descriptions of intrinsically disordered regions (IDRs). To accomplish this, we created the "IDRs" dataset with data from the DisProt database, as described in the methods chapter. The dataset contains input data for the fine-tuned T5 model, which is used to generate summaries. However, it lacks ground truth summaries since short descriptions are not available for IDRs. Therefore, we evaluated summary quality using the reference-free metric SummaC, specifically the $\text{SummaC}_{\text{Conv}}$ model. In the previous section, we discussed how this metric evaluates a different aspect of summary quality compared to reference-based metrics while still being useful for assessing the factual consistency of generated summaries with their input.

The distribution of $\text{SummaC}_{\text{Conv}}$ scores for the generated summaries is presented in Figure 3.15. The scores ranged from 0.196 to nearly 1.0, with an average of 0.452, indicating varying summary quality. To further analyze the model's performance, we examined three examples with low, medium, and high scores. These are presented in Table 3.6.

- **Low Score Example**: The $\text{SummaC}_{\text{Conv}}$ score for the IDR at positions 24 to 40 in the protein with ID 'P20849' was 0.196. Despite this low score, the summary correctly identified the region's carbohydrate-binding function, as inferred from its associated GO term. However, it lacked precision in describing the region's location, stating it was "found in hyaline cartilage and vitreous of the eye," instead of clarifying that it is part of a protein located in these tissues.
- **Medium Score Example**: The IDR at positions 327–382 of the protein with ID "P08050" received an evaluation score for its summary of 0.462. The summary was mostly correct, but it contained a hallucinated protein name, "GapJ," rather than the correct "Gap junction alpha-1 protein." This error implies that including protein names in the input might improve accuracy and reduce hallucinations. Furthermore, an error was found in the word "intrinsically," which was written as "intrainsically." We checked the rest of the generated summaries for this error and discovered that the model did not make the same mistake in all instances. In some cases, the term "intrinsically" was spelled correctly.

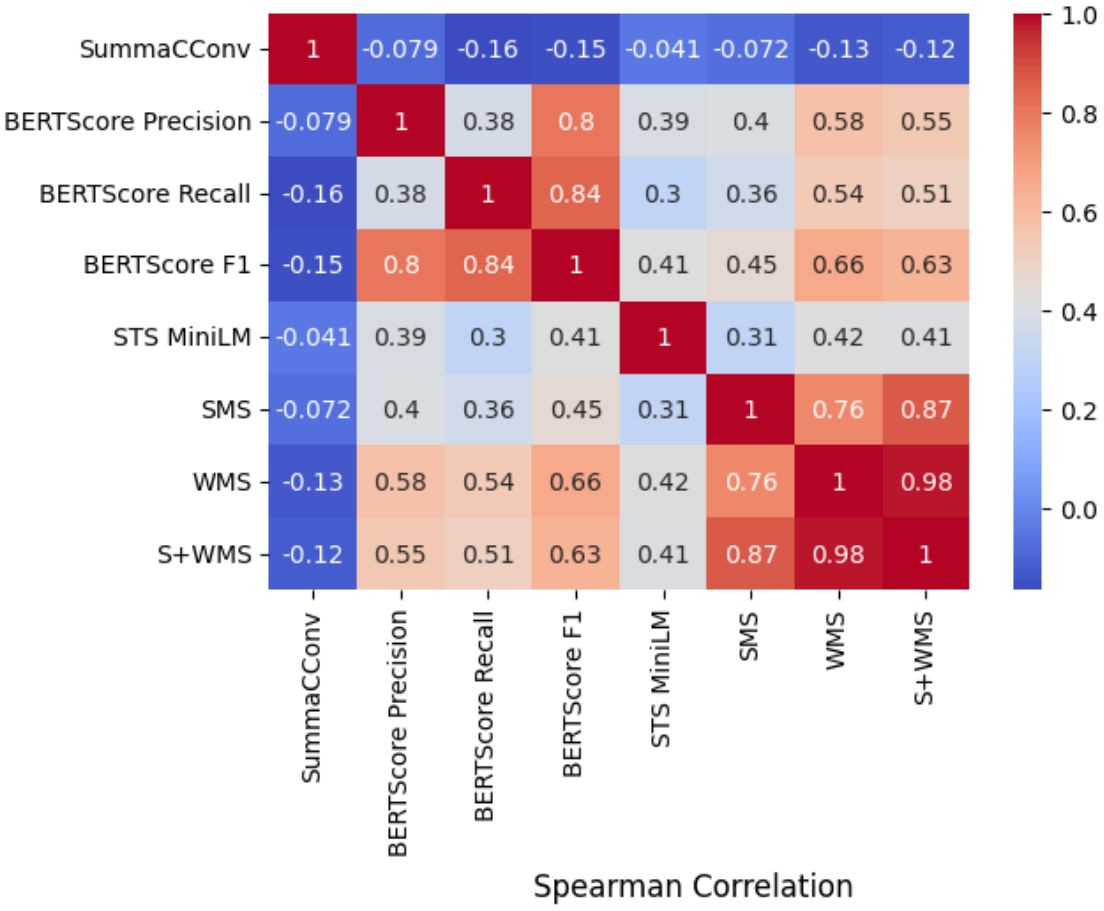

Figure 3.14: Spearman correlation heatmap showing the relationships between evaluation scores calculated with different metrics for summaries generated by the fine-tuned T5 model on the test set.

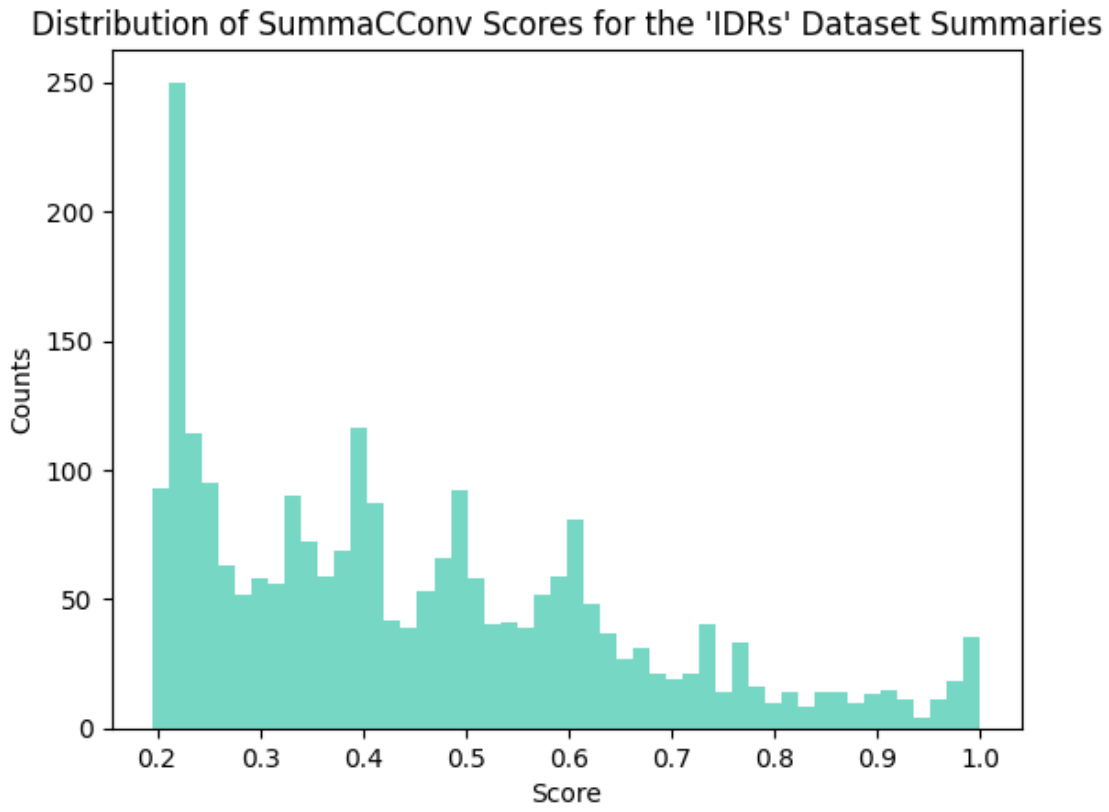

Figure 3.15: Histogram showing the distribution of $SummaC_{Conv}$ scores for the summaries generated by the fine-tuned T5 model on the ”IDRs” dataset.

- **High Score Example**: The summary for the IDR spanning positions 328-896 in protein with "Q05140" ID got a score of 0.967. The generated text was accurate and free of hallucinations. However, it relied more heavily on the protein's functional description, leaving out specific details from the associated GO terms that could have added depth to the resulting text. We further examined other examples from the dataset, and we found out that in some cases the model generated summaries based solely on the functional descriptions of the proteins without taking into account details from the GO term names and descriptions. The GO terms contain the most important information about the IDR's function and should not be left out.

Besides the three examples, we also examined additional summaries from the dataset. Another issue we discovered was incorrect information in the generated summaries about the C-terminus or N-terminus location of IDRs. When this information is available in the input, the model can use it correctly, as demonstrated in the second example in Table 3.6. However, if the input does not specify this, the model will be unable to accurately determine the region's location. To address this, one option is to supplement the dataset with positional data for the regions prior to fine-tuning for those dataset samples where this enrichment makes sense.

Many IDRs are associated with the "protein binding" GO term, indicating their binding activity. However, most input documents do not specify the protein to which these regions bind. Having this information in the generated summaries is very important. Therefore, enriching the input data with protein-protein interaction information would likely enhance the quality and completeness of the summaries.

Overall, our model successfully adapted to generate reasonable summaries for previously unseen data. However, there is still room for improvement, as we identified several issues with the fine-tuned T5 model, including incorrect protein names, hallucinations about region positions within proteins, misspellings, and omissions of important input details. In the following section, we will continue the discussion and propose directions for future improvements.

3.6 Future Work

Our approach is novel for automatically generating descriptions of protein sequence signatures, including IDRs. While our fine-tuned T5 model generates high-quality summaries in some cases, it also makes mistakes, indicating areas for improvement. This section proposes potential enhancements in multiple aspects, including dataset enrichments, alternative model choices, and other evaluation strategies.

We created three datasets of different complexity and analyzed the impact of dataset complexity on the quality of the generated summaries. The most complex dataset proved to be the best choice for fine-tuning our models. Despite incorporating GO term information, protein functional descriptions from the UniProtKB database, and organism details associated with specific InterPro entries, additional information could be beneficial.

Our findings revealed that the model occasionally attempted to include protein names in the output summaries, frequently hallucinating non-existent names due to the absence of this information in the input. Including protein names into the dataset could help link the summaries to accurate biological entities, reducing inaccuracies.

UniProtKB provides various types of general annotations (comments) for each protein, typically written in free text. We used only the comments on general protein functions in our most complex dataset. Adding other types of comments could help address gaps in the model's understanding. For example, including subunit structure annotations, which describe interactions with other proteins, could be particularly useful for IDRs associated

with GO terms indicating protein binding. Additional comments on domains, PTMs, and involvement in disease may further enrich the dataset.

Providing information about the location of a region (e.g., C-terminus or N-terminus), where appropriate, could reduce hallucinations related to positional details.

The T5 model demonstrated satisfactory performance in learning to generate descriptions, yet some inherent limitations should be considered. T5's maximum input token size of 1024 posed a challenge for examples with extensive annotations. Truncating input to meet this constraint likely resulted in the omission of important context, potentially degrading output quality. Exploring models with higher token limits could mitigate this issue. However, it is important to note that such models typically require more computational and storage resources for fine-tuning.

While our model had enough examples to learn the format of descriptions for protein families and domains, the absence of human-written descriptions for IDRs limited its ability to generalize to these regions. If possible, we recommend that expert biologists create a curated dataset by authoring at least a small number of descriptions specific to IDRs. Having such dataset may enable few-shot prompting with LLMs. Some LLMs, due to their strong in-context learning capabilities and large parameter counts, may have the potential to generate accurate IDR-specific descriptions without the need for additional fine-tuning. For example, some open-source LLM like Meta's Llama-3.1-70B*, or NVIDIA's Llama-3.1-Nemotron-70B-Instruct† could be used in this context. The input templates proposed in our study offer a solid foundation for designing effective prompts.

If the summaries generated by the few-shot prompting are not satisfactory, fine-tuning is still an option. Model can be fine-tuned exclusively on IDR data. That would potentially improve the precision of IDR-specific descriptions. Anyway, to achieve effective fine-tuning, a curated dataset containing a sufficient number of high-quality, ground-truth IDR summaries is essential.

Finally, evaluation remains a critical area for improvement. Automated metrics provide insights into textual similarity and fluency but often fail to capture domain-specific biological accuracy or subtle contextual errors. These limitations underscore the need for alternative evaluation strategies. Human evaluation could help fill these gaps by assessing biological accuracy and alignment with known protein signature functions in the generated descriptions. Establishing specific evaluation guidelines would allow for more systematic feedback and deeper insights into the model's strengths and weaknesses.

Biologist reviews would help determine whether the generated summaries are appropriate for use in a public database, with appropriate tags indicating that they are AI-generated. Such tags provide transparency for database users, allowing them to distinguish between human and AI-generated entries while maintaining accountability. While we can never be certain of the accuracy of AI-generated content without any review from humans, these methods significantly accelerate the curators' manual tasks. Instead of writing everything from scratch, experts can review and refine the suggested descriptions, saving time and effort.

*https://huggingface.co/meta-llama/Llama-3.1-70B

†https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Instruct

Custom ID	Input Text	Fine-tuned T5 Summary	SummaC Score
P20849_24_40	The Gene Ontology Terms for the given Intrinsically Disordered Region are: carbohydrate binding | Binding to a carbohydrate, which includes monosaccharides, oligosaccharides and polysaccharides as well as substances derived from monosaccharides by reduction of the carbonyl group (alditols), by oxidation of one or more hydroxy groups to afford the corresponding aldehydes, ketones, or carboxylic acids, or by replacement of one or more hydroxy group(s) by a hydrogen atom. Cyclitols are generally not regarded as carbohydrates. | molecular function It corresponds to Swiss-Prot proteins, with their functions and associated organisms listed below: ORGANISM: Homo sapiens (Human) FUNCTION: Structural component of hyaline cartilage and vitreous of the eye.	This entry represents the carbohydrate-binding region found in hyaline cartilage and vitreous of the eye.	0.196
P08050_327_382	The Gene Ontology Terms for the given Intrinsically Disordered Region are: flexible C-terminal tail | Intrinsically disordered region localized at the C-terminus of a protein. | disorder function It corresponds to Swiss-Prot proteins, with their functions and associated organisms listed below: ORGANISM: Rattus norvegicus (Rat) FUNCTION: Gap junction protein that acts as a regulator of bladder capacity. A gap junction consists of a cluster of closely packed pairs of transmembrane channels, the connexons, through which materials of low MW diffuse from one cell to a neighboring cell. Negative regulator of bladder functional capacity: acts by enhancing intercellular electrical and chemical transmission, thus sensitizing bladder muscles to cholinergic neural stimuli and causing them to contract. May play a role in cell growth inhibition through the regulation of NOV expression and localization. Plays an essential role in gap junction communication in the ventricles.	This entry represents the intrainsically disordered region found at the C-terminal end of Gap junction proteins, such as GapJ, which is a negative regulator of bladder functional capacity. It acts by enhancing intercellular electrical and chemical transmission, thus sensing bladder muscles to cholinergic neural stimuli and causing them to contract. It plays an essential role in gap junction communication in the ventricles.	0.464
Q05140_328_896	The Gene Ontology Terms for the given Intrinsically Disordered Region are: clathrin binding | Binding to a clathrin heavy or light chain, the main components of the coat of coated vesicles and coated pits, and which also occurs in synaptic vesicles. | molecular function clathrin coat assembly | The process that results in the assembly of clathrin triskelia into the ordered structure known as a clathrin cage. | biological process It corresponds to Swiss-Prot proteins, with their functions and associated organisms listed below: ORGANISMS: Rattus norvegicus (Rat), Homo sapiens (Human) FUNCTION: Adaptins are components of the adapter complexes which link clathrin to receptors in coated vesicles. Clathrin-associated protein complexes are believed to interact with the cytoplasmic tails of membrane proteins, leading to their selection and concentration. Binding of AP180 to clathrin triskelia induces their assembly into 60-70 nm coats. ORGANISM: Mus musculus (Mouse) FUNCTION: Adaptins are components of the adaptor complexes which link clathrin to receptors in coated vesicles. Clathrin-associated protein complexes are believed to interact with the cytoplasmic tails of membrane proteins, leading to their selection and concentration. Binding of AP180 to clathrin triskelia induces their assembly into 60-70 nm coats.	Adaptins are components of the adapter complexes which link clathrin to receptors in coated vesicles. AP180 binds to triskelia and induces their assembly into 60-70 nm coats.	0.967

Table 3.6: Examples of summaries generated by the fine-tuned T5 model for three samples of the "IDRs" dataset, along with their respective $\text{SummaC}_{\text{Conv}}$ scores.

4

Conclusion

In this thesis, we present a novel approach for automating the generation of short functional descriptions for protein sequence signatures, with a focus on IDRs. This study uses natural language generation models to bridge the gap between structured, machine-readable ontologies and human-readable summaries. These summaries could enhance researchers' access to functional information if integrated into protein databases in the future.

By constructing three datasets of varying complexity from InterPro entries and fine-tuning pre-trained models, including T5, GPT-2, and BioGPT, we demonstrated the capability of these models to generate high-quality summaries. Our results revealed that the complexity of the dataset significantly influences the quality of the generated summaries. The T5 model fine-tuned with the most complex dataset showed the best performance, benefiting from the inclusion of comprehensive protein descriptions and organism data. Moreover, the fine-tuned T5 model consistently outperformed the pre-trained model across all evaluation metrics, underscoring the importance of task-specific training.

A comparative analysis of the fine-tuned T5, GPT-2, and BioGPT models using both beam search and sampling decoding strategies revealed that the fine-tuned T5 model performed slightly better than the other models with the chosen hyperparameters for generation. However, this slight advantage was insufficient to conclude that it is clearly superior to the others. We also concluded that the choice of decoding strategy had a significant impact on the quality of the generated summaries. Furthermore, we looked into the impact of input features such as the number of GO terms and protein descriptions on summary quality and discovered that longer inputs may be slightly more problematic for the model, resulting in lower-quality summaries.

Importantly, our research demonstrated the model's ability to adapt to generating descriptions for IDRs, a category for which no explicit training data was available. The T5 model generated reasonable summaries for IDRs, but there were areas for improvement, such as dealing with the omission of critical information from the input and reducing hallucinations.

Future work can focus on several key areas to enhance the model's performance and applicability. Enriching the dataset with additional input features, such as protein names, subunit annotations, and positional details,

could help reduce inaccuracies and hallucinations. Exploring models that allow for larger token input sizes can address issues related to truncating input data. We believe that curated datasets for IDRs, provided by expert biologists, can significantly enhance the precision of the generated descriptions, and we advise collecting them if possible. Furthermore, combining human evaluation with automated metrics ensures the biological accuracy and contextual relevance of generated summaries. Setting specific guidelines for expert reviews can help provide systematic feedback and deeper insights into the model's performance.

Our research contributes significantly to the field of protein domains, families, and sequence features (including IDRs) annotation, providing a solid foundation for future work aimed at refining and expanding these methodologies. By automating the generation of functional descriptions, we hope to reduce the manual effort required of biological curators, thereby accelerating scientific discovery and enhancing the usability of protein databases.

In conclusion, our study demonstrates the feasibility and benefits of using natural language generation techniques to create concise, informative descriptions of protein signature functions. The successful application of these models to generate descriptions for IDRs marks a significant step forward in the automation of protein annotation tasks. Our findings show that fine-tuned language models have the potential to effectively generate functional descriptions, providing a foundation for further exploration of AI-driven approaches in protein annotation.

References

[1] T. Paysan-Lafosse, M. Blum, S. Chuguransky, T. Grego, B. L. Pinto, G. A. Salazar, M. L. Bileschi, P. Bork, A. Bridge, L. Colwell *et al.*, "Interpro in 2022," *Nucleic acids research*, vol. 51, no. D1, pp. D418–D427, 2023.

[2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[3] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[4] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, "Biogpt: generative pre-trained transformer for biomedical text generation and mining," *Briefings in bioinformatics*, vol. 23, no. 6, p. bbac409, 2022.

[5] K. M. Poluri, K. Gulati, and S. Sarkar, *Structural and Functional Properties of Proteins*. Singapore: Springer Singapore, 2021, pp. 1–60. [Online]. Available: https://doi.org/10.1007/978-981-16-1594-8_1

[6] R. Morris, K. A. Black, and E. J. Stollar, "Uncovering protein function: from classification to complexes," *Essays in Biochemistry*, vol. 66, no. 3, pp. 255–285, 2022.

[7] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[8] B. Lin, X. Luo, Y. Liu, and X. Jin, "A comprehensive review and comparison of existing computational methods for protein function prediction," *Briefings in Bioinformatics*, vol. 25, no. 4, 2024.

[9] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[10] M. Kumar, M. Gouw, S. Michael, H. Sámano-Sánchez, R. Pancsa, J. Glavina, A. Diakogianni, J. A. Valverde, D. Bukirova, J. Čalyševa *et al.*, "Elm—the eukaryotic linear motif resource in 2020," *Nucleic acids research*, vol. 48, no. D1, pp. D296–D306, 2020.

[11] D. Piovesan, A. M. Monzon, F. Quaglia, and S. C. Tosatto, "Databases for intrinsically disordered proteins," *Acta Crystallographica Section D: Structural Biology*, vol. 78, no. 2, pp. 144–151, 2022.

[12] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, "Molecular biology of the cell. 4th edn." 2003.

[13] D. Nelson and M. Cox, *Lehninger Principles of Biochemistry: 6th Edition*. Macmillan Learning, 2012. [Online]. Available: https://books.google.it/books?id=n9e1NAEACAAJ

[14] P. E. Wright and H. J. Dyson, "Intrinsically unstructured proteins: re-assessing the protein structure-function paradigm," *Journal of molecular biology*, vol. 293, no. 2, pp. 321–331, 1999.

[15] A. Cole and J. Eastoe, "Chapter 5 - peptides and proteins," in *Biochemistry and Oral Biology (Second Edition)*, A. Cole and J. Eastoe, Eds. Butterworth-Heinemann, 1988, pp. 44–70. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780723617518500120

[16] K. Tomii, "Protein properties," in *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds. Oxford: Academic Press, 2019, pp. 28–33. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128096338202665

[17] T. U. Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, no. D1, pp. D506–D515, 11 2018. [Online]. Available: https://doi.org/10.1093/nar/gky1049

[18] G. O. Consortium, "The gene ontology resource: 20 years and still going strong," *Nucleic acids research*, vol. 47, no. D1, pp. D330–D338, 2019.

[19] W. T. Clark and P. Radivojac, "Information-theoretic evaluation of predicted ontological annotations," *Bioinformatics*, vol. 29, no. 13, pp. i53–i61, 2013.

[20] A. Tomczak, J. M. Mortensen, R. Winnenburg, C. Liu, D. T. Alessi, V. Swamy, F. Vallania, S. Lofgren, W. Haynes, N. H. Shah *et al.*, "Interpretation of biological experiments changes with evolution of the gene ontology and its annotations," *Scientific reports*, vol. 8, no. 1, p. 5115, 2018.

[21] T. U. Consortium, "UniProt: the Universal Protein Knowledgebase in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, 11 2022. [Online]. Available: https://doi.org/10.1093/nar/gkac1052

[22] C. Vogel, M. Bashton, N. D. Kerrison, C. Chothia, and S. A. Teichmann, "Structure, function and evolution of multidomain proteins," *Current opinion in structural biology*, vol. 14, no. 2, pp. 208–216, 2004.

[23] N. Dawson, I. Sillitoe, R. L. Marsden, and C. A. Orengo, "The classification of protein domains," *Bioinformatics: Volume I: Data, Sequence Analysis, and Evolution*, pp. 137–164, 2017.

[24] C. P. Ponting and R. R. Russell, "The natural history of protein domains," *Annual review of biophysics and biomolecular structure*, vol. 31, no. 1, pp. 45–71, 2002.

[25] M. O. Dayhoff, "The origin and evolution of protein superfamilies." in *Federation Proceedings*, vol. 35, no. 10, 1976, pp. 2132–2138.

[26] H. Hegyi and M. Gerstein, "The relationship between protein structure and function: a comprehensive survey with application to the yeast genome," *Journal of molecular biology*, vol. 288, no. 1, pp. 147–164, 1999.

[27] C. A. Orengo and J. M. Thornton, "Protein families and their evolution—a structural perspective," *Annu. Rev. Biochem.*, vol. 74, no. 1, pp. 867–900, 2005.

[28] M. Chang, M. Dayhoff, R. Eck, and M. Sochard, "Atlas of protein sequence and structure," 1965.

[29] A. Heger and L. Holm, "Towards a covering set of protein family profiles," *Progress in Biophysics and Molecular Biology*, vol. 73, no. 5, pp. 321–337, 2000.

[30] C. H. Wu, H. Huang, L.-S. L. Yeh, and W. C. Barker, "Protein family classification and functional annotation," *Computational Biology and Chemistry*, vol. 27, no. 1, pp. 37–47, 2003.

[31] T. D. Bugg, *Introduction to enzyme and coenzyme chemistry*. John Wiley & Sons, 2012.

[32] S. Shanmugam, *Enzyme technology*. IK International Pvt Ltd, 2009.

[33] U.S. National Library of Medicine, "Binding site," https://www.nlm.nih.gov/mesh, n.d., medical Subject Headings (MeSH).

[34] Q. Zhong, X. Xiao, Y. Qiu, Z. Xu, C. Chen, B. Chong, X. Zhao, S. Hai, S. Li, Z. An *et al.*, "Protein posttranslational modifications in health and diseases: Functions, regulatory mechanisms, and therapeutic implications," *MedComm*, vol. 4, no. 3, p. e261, 2023.

[35] J. Heringa, "Detection of internal repeats: how common are they?" *Current Opinion in Structural Biology*, vol. 8, no. 3, pp. 338–345, 1998.

[36] M. A. Andrade, C. P. Ponting, T. J. Gibson, and P. Bork, "Homology-based method for identification of protein repeats using statistical significance estimates," *Journal of molecular biology*, vol. 298, no. 3, pp. 521–537, 2000.

[37] H. Satam, K. Joshi, U. Mangrolia, S. Waghoo, G. Zaidi, S. Rawool, R. P. Thakare, S. Banday, A. K. Mishra, G. Das *et al.*, "Next-generation sequencing technology: current trends and advancements," *Biology*, vol. 12, no. 7, p. 997, 2023.

[38] D. Piovesan, A. Del Conte, D. Clementel, A. M. Monzon, M. Bevilacqua, M. C. Aspromonte, J. A. Iserte, F. E. Orti, C. Marino-Buslje, and S. C. Tosatto, "Mobidb: 10 years of intrinsically disordered proteins," *Nucleic acids research*, vol. 51, no. D1, pp. D438–D444, 2023.

[39] H. J. Dyson and P. E. Wright, "Intrinsically unstructured proteins and their functions," *Nature reviews Molecular cell biology*, vol. 6, no. 3, pp. 197–208, 2005.

[40] R. Van Der Lee, M. Buljan, B. Lang, R. J. Weatheritt, G. W. Daughdrill, A. K. Dunker, M. Fuxreiter, J. Gough, J. Gsponer, D. T. Jones *et al.*, "Classification of intrinsically disordered regions and proteins," *Chemical reviews*, vol. 114, no. 13, pp. 6589–6631, 2014.

[41] A. S. Holehouse and B. B. Kragelund, "The molecular basis for cellular function of intrinsically disordered protein regions," *Nature Reviews Molecular Cell Biology*, vol. 25, no. 3, pp. 187–211, 2024.

[42] P. M. Kim, A. Sboner, Y. Xia, and M. Gerstein, "The role of disorder in interaction networks: a structural analysis," *Molecular systems biology*, vol. 4, no. 1, p. 179, 2008.

[43] J. Liu, N. B. Perumal, C. J. Oldfield, E. W. Su, V. N. Uversky, and A. K. Dunker, "Intrinsic disorder in transcription factors," *Biochemistry*, vol. 45, no. 22, pp. 6873–6888, 2006.

[44] C. A. Galea, Y. Wang, S. G. Sivakolundu, and R. W. Kriwacki, "Regulation of cell division by intrinsically unstructured proteins: intrinsic flexibility, modularity, and signaling conduits," *Biochemistry*, vol. 47, no. 29, pp. 7598–7609, 2008.

[45] M. M. Babu, R. van der Lee, N. S. de Groot, and J. Gsponer, "Intrinsically disordered proteins: regulation and disease," *Current opinion in structural biology*, vol. 21, no. 3, pp. 432–440, 2011.

[46] B. S. Schuster, G. L. Dignon, W. S. Tang, F. M. Kelley, A. K. Ranganath, C. N. Jahnke, A. G. Simpkins, R. M. Regy, D. A. Hammer, M. C. Good *et al.*, "Identifying sequence perturbations to an intrinsically disordered protein that determine its phase-separation behavior," *Proceedings of the National Academy of Sciences*, vol. 117, no. 21, pp. 11 421–11 431, 2020.

[47] M. C. Aspromonte, M. V. Nugnes, F. Quaglia, A. Bouharoua, S. C. Tosatto, and D. Piovesan, "Disprot in 2024: improving function annotation of intrinsically disordered proteins," *Nucleic Acids Research*, vol. 52, no. D1, pp. D434–D441, 2024.

[48] J. Roca-Martinez, T. Lazar, J. Gavalda-Garcia, D. Bickel, R. Pancsa, B. Dixit, K. Tzavella, P. Ramasamy, M. Sanchez-Fornaris, I. Grau *et al.*, "Challenges in describing the conformation and dynamics of proteins with ambiguous behavior," *Frontiers in molecular biosciences*, vol. 9, p. 959956, 2022.

[49] S. Vucetic, Z. Obradovic, V. Vacic, P. Radivojac, K. Peng, L. M. Iakoucheva, M. S. Cortese, J. D. Lawson, C. J. Brown, J. G. Sikes *et al.*, "Disprot: a database of protein disorder," *Bioinformatics*, vol. 21, no. 1, pp. 137–140, 2005.

[50] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[51] Anthropic, "Claude," 2024, large language model. [Online]. Available: https://www.anthropic.com/

[52] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[53] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, vol. 1. Minneapolis, Minnesota, 2019, p. 2.

[54] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[55] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[56] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[57] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–23, 2021.

[58] L. Zhuo, Z. Chi, M. Xu, H. Huang, H. Zheng, C. He, X.-L. Mao, and W. Zhang, "Protllm: An interleaved protein-language llm with protein-as-word pre-training," *arXiv preprint arXiv:2403.07920*, 2024.

[59] A. Radford, "Improving language understanding by generative pre-training," 2018.

[60] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[61] F. Stahlberg and B. Byrne, "On nmt search errors and model errors: Cat got your tongue?" *arXiv preprint arXiv:1908.10090*, 2019.

[62] G. Wiher, C. Meister, and R. Cotterell, "On decoding strategies for neural text generators," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 997–1012, 2022.

[63] E. Lloret, L. Plaza, and A. Aker, "The challenging task of summary evaluation: an overview," *Language Resources and Evaluation*, vol. 52, pp. 101–148, 2018.

[64] S. J. Giri, N. Ibtehaz, and D. Kihara, "Go2sum: generating human-readable functional summary of proteins from go terms," *npj Systems Biology and Applications*, vol. 10, no. 1, p. 29, 2024.

[65] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[67] M. Kilickaya, A. Erdem, N. Ikizler-Cinbis, and E. Erdem, "Re-evaluating automatic metrics for image captioning," *arXiv preprint arXiv:1612.07600*, 2016.

[68] J. Novikova, O. Dušek, A. C. Curry, and V. Rieser, "Why we need new evaluation metrics for nlg," *arXiv preprint arXiv:1707.06875*, 2017.

[69] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[70] E. Clark, A. Celikyilmaz, and N. A. Smith, "Sentence mover's similarity: Automatic evaluation for multi-sentence texts," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2748–2760.

[71] A. B. Sai, A. K. Mohankumar, and M. M. Khapra, "A survey of evaluation metrics used for nlg systems," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–39, 2022.

[72] T. Sun, J. He, X. Qiu, and X. Huang, "Bertscore is unfair: On social bias in language model-based metrics for text generation," *arXiv preprint arXiv:2210.07626*, 2022.

[73] W. Kryściński, N. S. Keskar, B. McCann, C. Xiong, and R. Socher, "Neural text summarization: A critical evaluation," *arXiv preprint arXiv:1908.08960*, 2019.

[74] P. Laban, T. Schnabel, P. N. Bennett, and M. A. Hearst, "Summac: Re-visiting nli-based models for inconsistency detection in summarization," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 163–177, 2022.

[75] T. Goyal and G. Durrett, "Evaluating factuality in generation with dependency-level entailment," *arXiv preprint arXiv:2010.05478*, 2020.

[76] A. R. Fabbri, C.-S. Wu, W. Liu, and C. Xiong, "Qafacteval: Improved qa-based factual consistency evaluation for summarization," *arXiv preprint arXiv:2112.08542*, 2021.

[77] T. Scialom, P.-A. Dray, P. Gallinari, S. Lamprier, B. Piwowarski, J. Staiano, and A. Wang, "Questeval: Summarization asks for fact-based evaluation," *arXiv preprint arXiv:2103.12693*, 2021.

[78] D. Deutsch, R. Dror, and D. Roth, "On the limitations of reference-free evaluations of generated text," *arXiv preprint arXiv:2210.12563*, 2022.

[79] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos, C. Xiong, Z. Z. Sun, R. Socher *et al.*, "Large language models generate functional protein sequences across diverse families," *Nature Biotechnology*, vol. 41, no. 8, pp. 1099–1106, 2023.

[80] S. Wang, R. You, Y. Liu, Y. Xiong, and S. Zhu, "Netgo 3.0: protein language model improves large-scale functional annotations," *Genomics, Proteomics & Bioinformatics*, vol. 21, no. 2, pp. 349–358, 2023.

[81] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[82] Q. Xie, J. A. Bishop, P. Tiwari, and S. Ananiadou, "Pre-trained language models with domain knowledge for biomedical extractive summarization," *Knowledge-Based Systems*, vol. 252, p. 109460, 2022.

[83] Y. Du, Q. Li, L. Wang, and Y. He, "Biomedical-domain pre-trained language model for extractive summarization," *Knowledge-Based Systems*, vol. 199, p. 105964, 2020.

[84] A. Gane, M. Bileschi, D. Dohan, E. Speretta, A. Héliou, L. Meng-Papaxanthos, H. Zellner, E. Brevdo, A. Parikh, M. Martin *et al.*, "Protnlm: model-based natural language protein annotation," *Preprint*, 2022.

[85] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. Sonnhammer *et al.*, "The pfam protein families database," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D138–D141, 2004.

[86] P. D. Thomas, M. J. Campbell, A. Kejariwal, H. Mi, B. Karlak, R. Daverman, K. Diemer, A. Muruganujan, and A. Narechania, "Panther: a library of protein families and subfamilies indexed by function," *Genome research*, vol. 13, no. 9, pp. 2129–2141, 2003.

[87] S. Yon Rhee, V. Wood, K. Dolinski, and S. Draghici, "Use and misuse of the gene ontology annotations," *Nature Reviews Genetics*, vol. 9, no. 7, pp. 509–515, 2008.

[88] D. Klopfenstein, L. Zhang, B. S. Pedersen, F. Ramírez, A. Warwick Vesztrocy, A. Naldi, C. J. Mungall, J. M. Yunes, O. Botvinnik, M. Weigel *et al.*, "Goatools: A python library for gene ontology analyses," *Scientific reports*, vol. 8, no. 1, pp. 1–17, 2018.

[89] F. Millstein, *Natural language processing with python: natural language processing using NLTK*. Frank Millstein, 2020.

[90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[91] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[92] X. Amatriain, A. Sankar, J. Bing, P. K. Bodigutla, T. J. Hazen, and M. Kazi, "Transformer models: an introduction and catalog," *arXiv preprint arXiv:2302.07730*, 2023.

[93] Y. Zhou, M. Keuper, and M. Fritz, "Balancing diversity and risk in llm sampling: How to select your method and parameter for open-ended text generation," *arXiv preprint arXiv:2408.13586*, 2024.

[94] Y. Jinnai, T. Morimura, and U. Honda, "On the depth between beam search and exhaustive search for text generation," *arXiv preprint arXiv:2308.13696*, 2023.

[95] K. Toutanova, C. Brockett, K. M. Tran, and S. Amershi, "A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs," in *EMNLP*, 2016.

[96] N. Reimers, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[97] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*. PMLR, 2015, pp. 957–966.

[98] W. Kryściński, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," *arXiv preprint arXiv:1910.12840*, 2019.

Acknowledgments

I would like to express my heartfelt gratitude to my mentors, Professor Damiano Piovesan and Mahta Mehdiabadi, for their invaluable guidance in defining the topic of my thesis, as well as their continuous support. I am also grateful to Professor Silvio Tosatto for welcoming me into the BioComputing UP Lab and to all my colleagues for creating such a friendly and collaborative working atmosphere.

A special thanks to my family, whose constant support, even from a distance, has given me the strength to overcome every challenge. Thank you for always believing in me. I am deeply grateful to my boyfriend, my adventure buddy, for his unconditional love and encouragement throughout this journey.

I would like to extend my thanks to all my friends and classmates I met during these two years in Padova for the wonderful memories we created together. I am also grateful to my lifelong friends, who have always been there for me, no matter where life has taken us.

Finally, I want to thank myself for my persistence and for never giving up. Well done for turning your dreams into reality, step by step.