DOCUMENTO PROVISÓRIO

Daniel Jorge
Bernardo Ferreira

Responder a questões de saúde do consumidor com linguagem não especializada

Answering consumer health questions with non-expert language

DOCUMENTO PROVISÓRIO

Daniel Jorge Bernardo Ferreira

Responder a questões de saúde do consumidor com linguagem não especializada

Answering consumer health questions with non-expert language

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Sérgio Matos, Professor associado com agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e Tiago Almeida, candidato a doutoramento em Engenharia Informática da Universidade de Aveiro.

o júri / the jury

presidente / president Por ser definido / to be defined

palavras-chave

Modelos de Linguagem, Geração de Texto, Simplificação de Texto Médico, Geração Controlável, Complexidade Textual, Informação de Saúde, Processamento de Linguagem Natural, Comunicação em Saúde

resumo

This work addresses the challenge of making health information accessible to diverse audiences through AI-powered text generation. The primary goal is to develop a controllable language model capable of generating health-related answers with adjustable complexity levels, making medical information understandable for both healthcare professionals and the general public. The research is structured around three main questions: (1) identifying the optimal combination of metrics to measure medical text complexity, (2) creating a high-quality dataset of health question-answer pairs annotated with complexity levels, and (3) developing a language model that can reliably generate health answers with adjustable complexity while maintaining accuracy and relevance. The proposed methodology involves analyzing existing readability metrics, creating a complexity-annotated dataset through a combination of expert knowledge and pre-trained language models, and exploring various approaches to controllable text generation, including fine-tuning, latent space manipulation, and reinforcement learning.

keywords

Language Models, Text Generation, Medical Text Simplification, Controllable Generation, Text Complexity, Health Information, Natural Language Processing, Healthcare Communication

abstract

Este trabalho aborda o desafio de tornar a informação de saúde acessível a diferentes públicos através da geração de texto baseada em Inteligência Artificial. O objetivo principal é desenvolver um modelo de linguagem controlável capaz de gerar respostas relacionadas com a saúde com níveis ajustáveis de complexidade, tornando a informação médica compreensível tanto para profissionais de saúde como para o público em geral. A investigação está estruturada em torno de três questões principais: (1) identificar a combinação ideal de métricas para medir a complexidade de textos médicos, (2) criar um conjunto de dados de elevada qualidade de pares de perguntas e respostas sobre saúde anotados com níveis de complexidade, e (3) desenvolver um modelo de linguagem que possa gerar de forma fiável respostas sobre saúde com complexidade ajustável, mantendo a precisão e relevância. A metodologia proposta envolve a análise de métricas de legibilidade existentes, a criação de um conjunto de dados anotado com complexidade através da combinação de conhecimento especializado e modelos de linguagem pré-treinados, e a exploração de várias abordagens para geração controlável de texto, incluindo fine-tuning, latent space manipulation, e reinforcement learning.

Table of contents

List of figures

List of tables

List of abbreviations

ARI . . . . . Automated Readability Index

CLI . . . . . Coleman-Liau Index

DCRS . . . . . Dale-Chall Readability Score

FKGL . . . . . Flesch-Kincaid Grade Level

GFI . . . . . Gunning Fog Index

QA . . . . . Question-Answering

RLHF . . . . . Reinforcement Learning with Human Feedback

SAMSA . . . . . Simplification Automatic evaluation Measure through Semantic Annotation

SARI . . . . . System Output Against References and Input

SMOG . . . . . Simple Measure of Gobbledygook

Chapter 1

Introduction

In recent years, there has been a growing demand for accessible and reliable health information online. Consumers increasingly turn to the internet to find answers to their health-related questions, seeking guidance on topics like symptoms, treatments, and preventive care. However, the complexity of medical information can be a significant barrier for many people. Most health content is written for a highly educated audience, using technical jargon and assuming a high level of background knowledge. This can lead to confusion, misunderstanding, and potentially even harmful decisions if people are unable to fully grasp the information they find. At the same time, oversimplifying medical information comes with its own risks. Leaving out important details for the sake of simplicity can also lead to misinterpretation and poor decision-making. The best solution would be to adjust how complex the information is based on who's reading it. A doctor should be able to read detailed technical explanations, while someone with no medical background should get clear, simple answers they can understand and use.

Artificial intelligence, particularly large language models (LLMs), have the potential to bridge this gap by dynamically generating health content at different complexity levels. Models like GPT-3 [1] and PaLM [2] can write clear text about many topics, including health. However, these models don't have reliable ways to control how complex their answers are. While we can ask them for "simple" or "detailed" answers, we can't be sure the text will match what different readers need. This is especially important for health information, where both overly simple and overly complex explanations could be harmful.

This work aims to create a system that combines AI language models with careful control over text complexity. This would make health information more useful for everyone, from medical experts to people with no health background. The system would ensure that simple explanations include all important information, while complex answers remain accurate when using advanced medical terms.

1.1 Objectives

The primary goal of this dissertation is to develop a controllable language model for consumer health Question-Answering (QA) that is capable of generating responses with adjustable complexity levels. This overarching goal is broken down into the following objectives:

- **Quantify Medical Text Complexity**: Identify a combination of metrics that effectively captures the complexity of medical text, addressing aspects such as specialized vocabulary, syntactic structures, and conceptual density.
- **Create a Complexity-Annotated Dataset**: Design and curate a high-quality dataset of health question-answer pairs annotated with complexity levels, facilitating the training and evaluation of the controllable QA model.
- **Develop a Controllable Language Model**: Design, implement, and evaluate a language model capable of generating health answers with user-specified complexity levels, ensuring factual accuracy, relevance, and fluency.

1.2 Document Outline

This dissertation is organized as follows:

- **Chapter 2: Background** reviews key concepts in language models, controlled text generation, and measuring language complexity. It explains different methods to control text generation and discusses metrics for measuring medical text complexity.
- **Chapter 3: Work Plan** presents the research questions, tasks, and timeline for developing a language model that can generate health answers with adjustable complexity. It describes the methodology for measuring text complexity, creating training data, and building the model.

Note on Document Length

This document has been condensed to meet the 25-page submission limit. Several sections, particularly in the latter parts of Chapter 2, have been abbreviated. Extended discussions of control methods, additional control methods and evaluation metrics, and detailed analyses are available if needed.

Chapter 2

Background

This chapter presents the foundational concepts and current research in controlled text generation. We begin with an overview of large language models and their text generation capabilities, followed by a detailed examination of controlled text generation methods. The chapter concludes with an analysis of language complexity measurement approaches, which are essential for evaluating text simplification systems.

2.1 Language Models

Language models are statistical models that learn to predict the probability distribution of words or tokens in a sequence. Modern neural language models, particularly those based on the Transformer architecture, process text through multiple layers of self-attention mechanisms that capture relationships between words at different positions in the sequence.

The text generation process in these models follows an autoregressive approach, where each token is predicted based on previously generated tokens. Given a sequence of tokens $X = \{x_1, x_2, ..., x_n\}$, the model computes the probability of the next token as:

$$P(X) = P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i|x_{<i}) \tag{2.1}$$

where $x_{<i}$ represents all tokens before position i. This formulation allows the model to generate text by iteratively sampling tokens from the predicted probability distribution.

Pre-trained language models learn these probability distributions through self-supervised training on large text corpora. During pre-training, models can acquire various types of knowledge:

- Syntactic patterns and grammatical rules
- Semantic relationships between words
- Domain-specific terminology and conventions

- Common facts and world knowledge

However, the standard generation process provides limited control over the attributes of the generated text. The model generates tokens based solely on learned probabilities, making it difficult to ensure specific characteristics in the output, such as maintaining a consistent style, following a particular format, or adhering to domain-specific constraints.

2.2 Controlled Text Generation

Controlled text generation addresses the challenge of guiding language models to produce text with specific desired properties while maintaining fluency and coherence [3–6]. Unlike standard text generation, controlled generation explicitly incorporates additional constraints or conditions into the generation process.

The task involves generating text that satisfies specified control conditions C while preserving the quality of the original language model output. When we incorporate these control conditions into the generation process, the probability distribution becomes:

$$P(X|C) = P(x_1, x_2, ..., x_n|C) = \prod_{i=1}^{n} p(x_i|x_{<i}, C) \tag{2.2}$$

Control conditions C can represent any definable text property and take various forms depending on the specific task requirements and application domain.

2.2.1 Control Conditions and Types

Control conditions in text generation serve different purposes based on the application requirements. These conditions broadly fall into three categories: semantic control, structural control, and lexical control.

Semantic Control

Semantic control focuses on abstract properties of the generated text:

- **Safety:** Text generation systems must avoid producing harmful, toxic, or biased content. This includes detecting and preventing discriminatory language, hate speech, or misleading information. For example, when generating dialogue responses, the system should avoid suggesting harmful actions or expressing biased views.

- **Sentiment:** Applications often need to control the emotional tone of generated text. A customer service chatbot might need to maintain a positive tone, while a news article generator should maintain neutral sentiment. The sentiment control extends beyond simple positive/negative classification to include fine-grained emotional states.

- **Topic:** Topic control ensures the generated text remains focused on specific subject matter. For instance, when generating scientific text, the system should maintain relevant terminology and concepts while avoiding unrelated topics. Topic control becomes particularly important in long-form text generation where maintaining thematic coherence is crucial.

Structural Control

Structural control manages the organization and format of generated text:

- **Format Specifications:** Many applications require text to follow specific formats. This includes generating poetry with particular rhyme schemes and meter, creating structured documents like academic papers or technical reports, or producing code with specific syntax requirements. The system must understand and maintain these format constraints throughout the generation process.

- **Document Organization:** Long-form text often requires specific organizational structures. This includes section ordering, paragraph breaks, and hierarchical relationships between different parts of the text. For example, a scientific paper generator needs to maintain standard sections like introduction, methods, results, and discussion.

- **Length Control:** Applications may need to generate text of specific lengths, from concise summaries to detailed explanations. Length control involves more than simple truncation - it requires generating complete, coherent text that naturally fits within the specified length constraints.

Lexical Control

Lexical control operates at the vocabulary level:

- **Keyword Inclusion:** Some applications require the generated text to incorporate specific keywords or phrases. This is common in search engine optimization, technical documentation, or domain-specific content generation. The challenge lies in naturally integrating these keywords while maintaining text fluency.

- **Vocabulary Constraints:** Text generation systems often need to restrict their vocabulary based on the target audience or domain. For example, text simplification systems must avoid complex terminology when generating content for general audiences. Similarly, technical writing might require using standardized terminology from a controlled vocabulary.

- **Term Consistency:** In technical or specialized writing, maintaining consistent terminology throughout the text is crucial. This includes using the same terms for

specific concepts and avoiding synonyms that might cause confusion.

2.2.2 Challenges in Control Integration

Integrating control conditions into text generation presents several technical challenges:

- **Control-Quality Trade-off:** Stronger control over text attributes often comes at the cost of reduced fluency or naturalness. Finding the right balance between control strength and text quality remains an ongoing challenge.
- **Multiple Constraint Interaction:** When multiple control conditions are applied simultaneously, they may conflict with each other. For example, maintaining a specific sentiment while including required keywords, or following a strict format while ensuring topic relevance.
- **Long-range Consistency:** As text length increases, maintaining consistent control becomes more difficult. Models may drift from the specified attributes or lose coherence over longer sequences.
- **Implicit Control Factors:** Some control aspects, like style or tone, are difficult to specify explicitly and may depend on subtle linguistic features that are hard to capture and manipulate.

2.2.3 Methods Overview

Methods for implementing controlled text generation can be divided into two main approaches: training-time methods and inference-time methods. This division reflects when control mechanisms are integrated into the text generation process [3, 7].

- **Training-time Methods:** These methods modify the language model during the training phase through various techniques:
  - Complete model retraining with control-specific objectives
  - Fine-tuning to adapt existing models
  - Training additional modules or adapters
  - Reinforcement learning with feedback signals
- **Inference-time Methods:** These methods guide text generation during inference without modifying the base model:
  - Prompt design and engineering
  - Guided decoding strategies

– Latent state manipulation

2.3 Training-time Methods

Training-time methods directly incorporate control mechanisms into the model's parameters or architecture. These methods aim to teach the model to recognize and respond to control signals during the training process.

2.3.1 Model Retraining

Complete model retraining is one of the earliest approaches to controlled text generation. This method involves training a language model from scratch with control-specific architectures or objectives.

The CTRL model [5] exemplifies this approach, introducing control codes to guide text generation. During training, each text segment in the dataset is paired with a control code that specifies attributes like style, domain, or topic. For example:

```
[Science] The study found significant correlations...
[Reviews Rating: 5.0] This product exceeded my expectations...
[Wikipedia] The Industrial Revolution began...
```

CTRL learns to associate these codes with specific text characteristics through natural co-occurrence during training. The model processes control codes as special tokens that influence the entire generation process. The training data preparation involves:

- Identifying relevant control attributes
- Creating consistent control code formats
- Collecting and labeling large-scale training data
- Ensuring balanced representation of different control codes

While CTRL demonstrates effective high-level control, the model cannot easily adapt to new control attributes without retraining, and the discrete nature of control codes limits the level of precision it can achieve.

The CoCon architecture [8] addresses these limitations by introducing a more flexible content-conditioning mechanism. Instead of relying on discrete control codes, CoCon embeds control signals directly into the model's hidden states. The architecture combines a base language model with a dedicated content-conditioning module that processes control signals. Integration layers then combine the conditioned and base representations to influence the generation process.

To train this architecture effectively, CoCon implements multiple complementary loss functions:

- **Primary Reconstruction Loss ($\mathcal{L}_{recon}$):** Ensures the model can accurately generate text that incorporates the given control signals.
- **Null Content Loss ($\mathcal{L}_{null}$):** Helps the model maintain robust performance even when control signals are absent, which is important for practical applications.
- **Cycle Reconstruction Loss ($\mathcal{L}_{cycle}$):** Promotes consistency in how control signals are applied. It ensures that if we extract control signals from a generated text and use them to condition another generation, we obtain similar results.
- **Adversarial Loss ($\mathcal{L}_{adv}$):** Improves the naturalness of the generated text by training the model to produce outputs that are indistinguishable from human-written text while maintaining the desired control attributes.

The total loss function for training the CoCon model is a weighted combination of these four components:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda_1 \mathcal{L}_{null} + \lambda_2 \mathcal{L}_{cycle} + \lambda_3 \mathcal{L}_{adv} \tag{2.3}$$

where the λ parameters balance the contribution of each loss term.

This multi-loss training strategy ensures that the model learns to balance control requirements with text fluency and coherence. Furthermore, by keeping the original language model intact and only adding a lightweight content-conditioning module, CoCon largely preserves the pre-trained knowledge and capabilities of the base model.

For scenarios requiring precise lexical control, the POINTER model [9] introduces a fundamentally different approach based on insertion-based generation. Unlike traditional left-to-right generation models that predict one token at a time in sequence, POINTER can insert tokens at any position in the sequence. This makes it particularly well-suited for ensuring specific keywords or phrases appear in the generated text.

POINTER operates through a progressive refinement process. The generation begins with an initial sequence containing only the required lexical constraints (i.e., the keywords or phrases that must appear in the final text). The model then iteratively inserts new tokens between existing ones, gradually building up the complete text. At each step, a learned policy determines both which tokens to insert and where to place them, considering the surrounding context and the overall coherence of the sequence.

This insertion-based approach provides several advantages for lexically constrained generation. It guarantees that required keywords will appear in the output since they form the starting point of generation. The ability to insert tokens at any position allows for more flexible text construction compared to strict left-to-right generation. However, this flexibility comes with computational costs, as each insertion operation requires evaluating multiple possible positions and tokens. The process can become particularly intensive for longer sequences where many insertions are needed to complete the text.

2.3.2 Fine-tuning

Fine-tuning adapts a pre-trained language model to handle controlled text generation by adjusting its parameters using specialized datasets or training objectives. This approach preserves the general language generation capabilities of the pre-trained model while improving its ability to respond to specific control signals.

The fine-tuning process modifies the model parameters according to:

$$\Theta^* = \Theta + \Delta\Theta \tag{2.4}$$

where Θ represents the original parameters of the pre-trained model, and $\Delta\Theta$ represents the updates learned during fine-tuning. The parameter updates are computed by minimizing a task-specific loss:

$$\Delta\Theta = \arg\min_{\Theta} \mathcal{L}(D_{control}, f(X;\Theta)) \tag{2.5}$$

where $D_{control}$ is a dataset designed for the control task, and $f(X;\Theta)$ represents the model's output for input X.

Adapter-Based Fine-tuning

Adapter-based fine-tuning introduces specialized neural modules into the pre-trained model while keeping its original parameters frozen. This reduces the risk of catastrophic forgetting, where fine-tuning causes the model to lose previously learned knowledge.

Auxiliary Tuning [10] adds a separate auxiliary model alongside the pre-trained language model. The auxiliary model processes both the input text and control signals, producing logits that are combined with the base model's output through a softmax operation:

$$P(y|x,C) = \text{softmax}(f_{LM}(x) + f_{AUX}(x,C)) \tag{2.6}$$

where f_{LM} represents the frozen pre-trained model and f_{AUX} is the trainable auxiliary model. Intuitively, the auxiliary model learns the residual logits needed to shift the probability distribution of the pre-trained model towards the target distribution. To further improve training efficiency, the lower layers of the pre-trained model can also be used as a feature extractor for the auxiliary model inputs. There are no constraints on the auxiliary model architecture, except that it must output logits over the same vocabulary as the original model. It is typically much smaller than the pre-trained model (e.g., a few Transformer layers) and can be trained on a small amount of data.

DisCup [11] combines discriminator guidance with prompt-tuning. The training objective includes both likelihood and unlikelihood terms:

$$\mathcal{L}_{total} = \sum_{i=1}^{|D|} \sum_{t=1}^{|x^{(i)}|} \mathcal{L}_{like}(x_t^{(i)}) + \mathcal{L}_{unlike}(x_t^{(i)}) \tag{2.7}$$

For each training step, the base model generates candidate tokens which the discriminator scores based on their alignment with desired attributes. $\mathcal{L}_{like}$ maximizes the probability of positively-scored tokens, while $\mathcal{L}_{unlike}$ minimizes the probability of negatively-scored ones. The unlikelihood training helps prevent the model from learning spurious correlations present in training data. Furthermore, by operating on self-generated candidates rather than ground-truth tokens, DisCup maintains output diversity while improving attribute control.

LiFi [12] proposes an different approach using lightweight adapters guided by attribute classifiers. The adapters are added to each transformer layer while keeping the base model frozen:

$$h' = h + \text{Adapter}(h, c) \tag{2.8}$$

where h is the layer's hidden state and c is the control signal from the classifier. The adapters use a bottleneck architecture with reduction factors of 16 and 4 for feedforward and attention modules respectively, adding only 0.04% parameters to the base model.

During generation, multiple adapters can be combined through a weighted fusion mechanism:

$$h'_t = h_t + \sum_{i=1}^{k} w_i \text{Adapter}_i(h_t, c_i) \tag{2.9}$$

where w_i are learnable weights balancing different control signals, and k is the number of adapters. This allows LiFi to handle multiple control attributes while maintaining computational efficiency.

Data-Driven Fine-Tuning

Data-driven methods focus on creating better datasets to teach language models about control through fine-tuning.

FLAN [13] showed that turning NLP tasks into natural language instructions helps models learn new tasks without examples. For instance, instead of training a model to summarize text, FLAN trains it to follow instructions like "Summarize the text" or "Write a positive review about the product". This helps the model understand what it needs to do through clear directions, enabling zero-shot generalization to new tasks.

Building on this idea, InstructCTG [14] applies instruction-based fine-tuning specifically to controlled text generation. It turns control constraints into natural instructions and uses them to fine-tune language models. This makes the control process more intuitive, as the model learns to follow human-like directions rather than abstract control signals.

CHRT [15] uses contrastive learning to control multiple attributes at once. It works by comparing texts that have different attributes (e.g., positive vs negative sentiment) and learning how these differences appear in the model's hidden layers. This helps the model

understand how to change its output to match desired attributes without changing its basic structure. CHRT combines two loss functions to balance attribute control and text quality:

- **Contrastive Loss ($\mathcal{L}_c$):** Uses triplet loss to guide the transformed hidden representations towards a model fine-tuned on desired attributes and away from one fine-tuned on undesired attributes. For example, when controlling toxicity, it pushes the representations closer to a non-toxic model and farther from a toxic one.
- **Preservation Loss ($\mathcal{L}_p$):** Maintains the model's original language capabilities by minimizing the distance between the transformed representations and the base model's representations.

The total loss is a weighted combination of these two omponents:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_p + (1 - \lambda)\mathcal{L}_c \tag{2.10}$$

where λ balances between preserving the original model's capabilities and achieving strong attribute control. One advantage of CHRT is that it can control multiple attributes by combining different transformation blocks during inference, without needing to retrain the model.

2.3.3 Reinforcement Learning

Reinforcement learning methods optimize text generation by providing feedback signals that indicate how well the generated text satisfies the control objectives. These methods can handle complex, non-differentiable control criteria and adapt to human preferences. The RL optimization adjusts model parameters Θ according to:

$$\Theta^* = \Theta + \alpha \nabla_\Theta \mathbb{E}_{\pi_\Theta}[R(X)] \tag{2.11}$$

where α is the learning rate, π_Θ is the policy defined by the model parameters, and $R(X)$ is the reward for generated text X. Since the exact reward function is often unknown, in practice, RL methods like REINFORCE [16] use the policy gradient to estimate the gradient of the expected reward:

$$\nabla_\Theta \mathbb{E}_{\pi_\Theta}[R(X)] \approx \sum_{t=1}^{T} \nabla_\Theta \log P(x_t | x_{<t}, C) R(X) \tag{2.12}$$

RL methods fall into two categories based on how they get their feedback: from humans or from automated systems.

Human Feedback Methods

Human feedback methods use human evaluators to provide reward signals for the generated text. These methods are particularly useful for subjective control attributes that are hard to define algorithmically.

Reinforcement Learning with Human Feedback (RLHF) [17] combines human feedback with reinforcement learning to control text summarization. The method consists of three stages: collecting human feedback, training a reward model, and optimizing the language model using RL.

The first stage creates a dataset of human preferences by presenting annotators with pairs of model outputs and asking them to select the better summary. This process generates a dataset D of preferences (x, y_0, y_1, i), where x is the input text, y_0 and y_1 are the two generated summaries, and i indicates the preferred summary.

The reward model $r_\theta(x, y)$ is then trained to predict human preferences:

$$\mathcal{L}_{reward} = -\mathbb{E}_{(x,y_0,y_1,i)\sim D}[\log(\sigma(r_\theta(x, y_i) - r_\theta(x, y_{1-i}))] \tag{2.13}$$

This loss function encourages the reward model to assign higher scores to summaries preferred by humans.

Finally, the language model is fine-tuned using Proximal Policy Optimization (PPO) to maximize the predicted reward while staying close to the original model's behavior through a KL penalty:

$$R(x, y) = r_\theta(x, y) - \beta \ \mathrm{KL}(\pi_{RL}, \pi_{base}) \tag{2.14}$$

where β balances between maximizing the predicted reward and maintaining the original model's behavior.

Safe RLHF [18] extends RLHF to address a fundamental tension in language model alignment: balancing helpfulness with harmlessness. Traditional RLHF approaches often struggle when these objectives conflict, as a model being maximally helpful (e.g., providing detailed information about dangerous topics) might compromise safety. Safe RLHF decouples these two objectives by introducing a cost model $C_\psi(y, x)$ that penalizes harmful content. The optimization objective becomes:

$$\begin{aligned} \text{maximize}_\theta \quad & \mathbb{E}_{x,y}[R_\phi(y, x)] \\ \text{subject to} \quad & C_\psi(y, x) \leq 0 \end{aligned} \tag{2.15}$$

where R_ϕ is the reward model and C_ψ is the cost model. This constraint ensures generated text remains helpful while avoiding harmful content.

The problem is solved using Lagrangian relaxation:

$$\min_{\lambda \geq 0} \max_{\theta} [-J_R(\theta) + \lambda J_C(\theta)] \tag{2.16}$$

where J_R and J_C are the expected reward and cost, and λ is the Lagrange multiplier that adjusts the trade-off between the two objectives. When the model becomes less safe, λ increases to prioritize harmlessness, and when safety improves, λ decreases to focus more on helpfulness.

Automated Feedback Methods

Automated feedback methods guide language model training using predefined reward functions rather than human feedback. These methods scale better since they don't require collecting human preferences.

Distributional approaches like GDC [19] treat controlled generation as a constrained optimization problem. The goal is to find a text distribution that satisfies the control requirements while staying close to the original model distribution. This prevents the model from generating unnatural text when trying to meet control conditions. GDC solves this by finding:

$$P(x) = \arg\min_{c \in C} D_{KL}(c|a) \tag{2.17}$$

where P is the target distribution, C contains distributions meeting the constraints, and a is the original model distribution. The solution takes an energy-based form:

$$P(x) \propto a(x) e^{\sum_i \lambda_i \phi_i(x)} \tag{2.18}$$

Here $\phi_i(x)$ are feature functions encoding control conditions and λ_i are learned parameters. GDC learns these parameters through importance sampling, making it more efficient than direct optimization.

A major challenge in reinforcement learning for text generation is the sparse reward problem, where models only receive feedback after generating complete sequences. DRL [20] addresses this by providing dense rewards at each generation step. The method combines three types of token-level rewards:

$$r_t = \lambda_s r_t^s + \lambda_c r_t^c + \lambda_f r_t^f \tag{2.19}$$

The style reward r_t^s uses attention scores from a classifier to identify style-relevant tokens. Content preservation reward r_t^c measures n-gram overlap with the input. Fluency reward r_t^f comes from a language model to maintain natural text. This dense feedback helps the model learn more effectively than with sparse sentence-level rewards.

For medical text simplification, TESLEA [21] combines fine-tuning and reinforcement learning. After initial training on paired medical reviews, the method optimizes three rewards:

- **Relevance Reward (R_{cosine}):** Uses BioSentVec embeddings [22] to measure semantic similarity between original and simplified text, preserving medical meaning.

- **Readability Reward (R_{Flesch}):** Encourages text to reach a target grade level using Flesch-Kincaid readability scores.
- **Lexical Simplicity Reward ($R_{lexical}$):** Promotes common and accessible vocabulary based on word frequency statistics following Zipf's law.

The total reward combines these components:

$$R_{total} = \alpha R_{cosine} + \beta R_{Flesch} + \gamma R_{lexical} \tag{2.20}$$

TESLEA uses Self-Critical Sequence Training [23], generating two versions of each simplification through greedy and sampling-based decoding. The final loss balances reinforcement learning against MLE to prevent drift:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_{RL} + (1 - \lambda)\mathcal{L}_{MLE} \tag{2.21}$$

TESLEA made medical texts significantly easier to read, lowering the FKGL by 2.5 points while maintaining high ROUGE and SARI scores. When evaluated by domain experts, over 70% agreed on the quality of the simplifications across multiple dimensions including fluency, coherence, and factual accuracy. The outputs from TESLEA were consistently shorter than those produced by baseline models and the input paragraphs themselves, suggesting it learned to remove unnecessary technical details while retaining the most relevant content.

2.4 Inference-time Methods

Inference-time methods change how language models generate text without modifying their parameters. This makes them flexible and practical, especially when working with large models that are expensive to retrain.

2.4.1 Prompt Engineering

Prompt engineering controls text generation by providing specific instructions or cues to the language model. These can be either hard prompts (natural language instructions) or soft prompts (trainable vectors).

Hard prompts are direct text instructions that tell the model what to do. For example:

```
Write a medical text about diabetes using simple language:
Explain the risks of smoking in a positive tone:
```

While simple to use, hard prompts can be unpredictable, small changes in wording can lead to very different outputs. To address this, researchers developed methods to find effective prompts automatically. AutoPrompt [24] uses gradient-based search to find the best trigger words for a given task. It starts with a template containing [MASK] tokens and replaces them with words that maximize the probability of desired outputs.

Soft prompts offer more reliable control by using trainable vectors instead of words. Prefix-Tuning [25] adds these vectors before each layer of the model, while keeping the model's parameters fixed. The vectors learn to guide the model toward specific outputs during training. When tested on table-to-text generation and summarization, Prefix-Tuning matched the performance of full model fine-tuning while changing only 0.1%-2% of the parameters.

P-tuning [26] improved soft prompting by processing the prompt vectors through a bidirectional LSTM network. This made training more stable and worked better for both classification and generation tasks. The method was particularly good at handling new situations with few examples.

2.4.2 Guided Decoding

Guided decoding is a technique used during the decoding process of a language model to bias the generated text towards the desired control attributes. It works by manipulating the logits or probability distribution of the model to favor tokens that align with the control conditions. This is often done using classifiers or reward models that provide additional supervision signals to guide the generation process.

PPLM [6] was one of the first methods to show this was possible. It uses simple attribute models (e.g., sentiment classifiers) to adjust the internal states of GPT-2. This lets it control attributes like topic and tone without changing the base model. However, because it needs multiple passes through both models for each word, generation is quite slow.

GeDi [27] made this process faster by using small language models trained on specific attributes instead of classifiers. For example, one model trained on positive text and another on negative text. During generation, GeDi compares how likely each potential next word is under both models to decide which words to favor.

DExperts [28] simplified this further by directly combining predictions from three models: a base model, an "expert" trained on desired text (like non-toxic), and an "anti-expert" trained on undesired text (like toxic). This worked well for making GPT-2 and GPT-3's text less toxic while staying natural. Small expert models (125M parameters) could successfully guide much larger base models (1.5B parameters).

Mix and Match [29] took a different approach by treating text generation as a sampling problem. It combines scores from different models (for fluency, topic, safety, etc.) into a single energy function. The method then uses Metropolis-Hastings sampling to find text that balances all these aspects well. While this produced good results, the sampling process made generation slower than other methods.

2.4.3 Latent Space Manipulation

Latent space manipulation controls text by adjusting the hidden states inside the language model. These hidden states form a high-dimensional space where similar words

and concepts are close together. By moving through this space in specific directions, we can change the attributes of generated text.

ICV [30] finds these directions by comparing how the model processes examples with different attributes. For instance, comparing the hidden states when processing positive versus negative text reveals a "sentiment direction". Adding or subtracting along this direction during generation then makes text more positive or negative.

ActAdd [31] showed that even simple changes to these hidden states can effectively control generation. The method first identifies which parts of the hidden states correspond to specific attributes. It then adjusts these values during generation, similar to turning a dial to control different aspects of the text.

MIRACLE [32] handles multiple attributes at once by mapping text into a special latent space designed to keep different attributes separate. This helps prevent attributes from interfering with each other, allowing more precise control over multiple aspects of the generated text.

2.5 Evaluation Metrics

2.5.1 Traditional Readability Metrics

Readability formulas estimate text difficulty using surface-level features such as average sentence length, word length, and proportion of complex words. These metrics help evaluate many types of medical content, from patient education materials for conditions like nocturnal enuresis [33] and bariatric surgery [34], to discharge instructions [35] and critical care information [36].

Flesch-Kincaid Grade Level (FKGL) estimates the U.S. grade level needed to understand a text:

$$\text{FKGL} = 0.39 \times \left(\frac{\text{words}}{\text{sentences}}\right) + 11.8 \times \left(\frac{\text{syllables}}{\text{words}}\right) - 15.59 \tag{2.22}$$

FKGL scores typically range from 0 to 18, with higher scores indicating more difficult text. A score of 9.2 suggests text suitable for ninth-grade students, while scores above 12 indicate college-level or specialized content.

Simple Measure of Gobbledygook (SMOG) calculates text difficulty based on polysyllabic words in a 30-sentence sample:

$$\text{SMOG} = 1.0430 \times \sqrt{\text{polysyllables} \times \left(\frac{30}{\text{sentences}}\right)} + 3.1291 \tag{2.23}$$

Like FKGL, SMOG scores are interpreted as U.S. grade levels. SMOG has gained wide adoption in healthcare settings due to its relative ease of use and focus on vocabulary complexity.

Dale-Chall Readability Score (DCRS) measures text difficulty by calculating the percentage of words not found in a list of 3,000 familiar words:

$$\text{DCRS} = 0.1579 \times \left(\frac{\text{difficult words}}{\text{total words}} \times 100 \right) + 0.0496 \times \left(\frac{\text{total words}}{\text{total sentences}} \right) \tag{2.24}$$

If the percentage of difficult words exceeds 5%, an additional constant of 3.6365 is added to the raw score to get the final DCRS. Scores range from 4.9 or below for easily understood text to 10 or above for very challenging text.

Other commonly used metrics include the **Automated Readability Index (ARI)**, **Coleman-Liau Index (CLI)**, **Gunning Fog Index (GFI)**, and **Linsear Write Formula**. hile the specific calculations differ, they all aim to estimate text difficulty based on factors like word length, sentence length, and syllable counts.

Traditional readability formulas face several limitations when applied to medical texts:

- They only examine surface-level features, missing deeper aspects of conceptual complexity in medical information [37–39].
- They treat the text as a "bag of words", ignoring features like information density, organization, coherence, and syntax.
- They do not consider the cognitive load imposed by specific terms [40]. Short terms like "polyp" and "apnea" may be challenging for lay readers, while longer words like "gastrointestinal" and "cardiovascular" may be more familiar in a medical context.
- They were developed using general reading materials, often geared towards children's education levels. The assumptions and thresholds used in these formulas may not generalize well to the specialized language and adult literacy levels of medical texts [37].

Despite these limitations, readability formulas provide a quantitative starting point for improving medical text clarity. Experts generally recommend using them alongside other evaluation approaches, combining multiple metrics with expert review and user testing [41, 42].

2.5.2 Reference-based Metrics

Reference-based metrics compare generated text against human-written reference simplifications. These metrics help measure how well the system's output matches examples created by domain experts or crowdsourced annotators.

System Output Against References and Input (SARI) [43] specifically targets text simplification quality by comparing n-grams in the candidate and reference texts. For a given word sequence, SARI calculates:

$$\text{SARI} = \frac{F_{add} + F_{keep} + F_{del}}{3} \tag{2.25}$$

where F_{add}, F_{keep}, and F_{del} are F1 scores for added, kept, and deleted n-grams compared to the reference text. This metric is particularly relevant for medical text simplification as it captures both readability improvements and content preservation [44].

BERTScore [45] compares generated text against references using contextual embeddings from BERT models. For a candidate token x_i and reference token $\hat{x}_j$, it computes their similarity using cosine similarity of their BERT embeddings. The metric combines these token-level similarities into precision and recall scores:

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j \tag{2.26}$$

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^T \hat{x}_j \tag{2.27}$$

The final BERTScore is an F1 measure that balances precision and recall:

$$F_{BERT} = \frac{2(P_{BERT} \cdot R_{BERT})}{P_{BERT} + R_{BERT}} \tag{2.28}$$

This approach catches semantic similarities that n-gram metrics miss, since BERT embeddings can match related words like "car" and "vehicle". It also handles word reordering better since matching is done at the token level. BERTScore has shown the highest correlations with human judgments of simplicity and meaning preservation compared to other metrics, particularly when evaluating the outputs of neural sequence-to-sequence models [46], though it is unclear how well it generalizes to the medical domain.

2.5.3 Reference-free Metrics

Reference-free metrics evaluate text without relying on human-written reference simplifications. This makes them appealing because collecting high-quality human reference simplifications is time-consuming, expensive, and may not always be feasible [46–48].

Simplification Automatic evaluation Measure through Semantic Annotation (SAMSA) [49] measures structural simplification by analyzing how well complex sentences are broken down into simpler ones. It uses semantic parsing to identify the main ideas in a sentence and checks if each appears in its own simple sentence. For a simplified text S containing n semantic units, SAMSA calculates:

$$\text{SAMSA} = \frac{\text{number of properly allocated semantic units}}{n} \tag{2.29}$$

This metric is especially relevant for medical text, where breaking down complex medical concepts into digestible units improves reader understanding.

2.5.4 LLM-based Metrics

LLM-based metrics use large language models to evaluate text quality. Unlike traditional metrics that count words or measure sentence length, these metrics can understand

meaning and assess writing style.

G-Eval [50] uses GPT-4 to score text on four aspects:

- **Coherence** (1-5): How well ideas connect and flow
- **Consistency** (1-5): Whether all facts match the source text
- **Fluency** (1-3): Grammar and sentence quality
- **Relevance** (1-5): Whether important information is kept

G-Eval samples multiple scores for each aspect using temperature sampling, then combines them with a probability weighting:

$$\text{Score} = \sum_{i=1}^{n} p_i s_i \tag{2.30}$$

where p_i is the probability of each score, s_i is the score value, and n is the number of samples.

G-Eval prompts GPT-4 to evaluate each aspect separately, asking for step-by-step reasoning before giving a score. While originally designed for summarization tasks, its framework could be adapted for evaluating medical text simplification, though it may require additional fine-tuning on medical data.

Chapter 3

Work Plan

This chapter presents the work plan for my thesis. The goal is to develop a language model that generates answers to consumer health questions with adjustable complexity levels. The model should be able to produce simple and concise answers for non-experts, as well as detailed and technical responses for healthcare professionals.

This plan is structured around three core research questions (RQ), with detailed tasks and a timeline spanning from January to June 2025. The timeline is provisional and may change based on progress or unforeseen challenges.

3.1 Research Questions

To guide the development of the controllable health QA model, we define the following research questions:

RQ1: What combination of metrics best captures the complexity of medical text?

To generate answers at different complexity levels, we first need to figure out how to measure text complexity in the medical domain. There are existing readability formulas like FKGL, but they do not capture all the aspects that make medical text hard to understand, like specialized vocabulary and complex sentence structures. The goal is to find the combination of metrics that does the best job distinguishing simple and complex medical writing in a way that aligns with human judgment.

Task 1.1: Literature Review:

- Conduct a thorough review of existing literature on readability metrics and complexity measures tailored for health-related text.
- Identify the most promising metrics and measures to be used in the project.

Task 1.2: Dataset Preparation

- Search for and acquire a dataset containing paired examples of complex and simplified health articles or documents.
- Preprocess the dataset, ensuring it is clean, well-structured, and ready for analysis.

Task 1.3: Metric Selection and Evaluation

- Implement the selected readability metrics and complexity measures identified in Task 1.1.
- Calculate the complexity scores for each text pair in the preprocessed dataset.
- Select the combination of metrics that best distinguishes between complex and simple texts based on the results of the Kullback-Leibler (KL) divergence, clustering (e.g., K-means), and classification (e.g., logistic regression) experiments.

Milestone 1: Development of a robust method for quantifying the complexity of health-related text.

RQ2: How do we create good training dataset for complexity controlled text generation?

To train a model to output answers at different complexity levels, we first need a dataset that exhibits this type of variance. Since there aren't any datasets like this, we'll need to create our own by collecting medical text from various sources and annotating it with the set of metrics identified in RQ1.

Task 2.1: Data Collection

- Identify a suitable dataset of expert-level health question-answer pairs.
- Preprocess the dataset, ensuring it is clean, well-structured, and ready for use.

Task 2.2: Answer generation using pre-trained models

- Select and set up a pre-trained language model, such as BioBERT or SciBERT, for answer generation.
- Feed the question and reference answer into the chosen pre-trained model and prompt it to generate several alternative answers, ranging from very simple to highly complex.

Task 2.3: Dataset Annotation

- Apply the complexity quantification method developed in RQ1 to the generated answers.
- Create a complexity-stratified dataset by labeling each generated answer with its corresponding complexity level.

Task 2.4: Answer Validation

- Validate the generated answers to ensure they are factually accurate and relevant to the original questions.
- Remove any low-quality or irrelevant answers from the dataset.

Milestone 2: Curation of a high-quality dataset of health question-answer pairs with annotated complexity levels.

RQ3: How can we design a language model that reliably generates health answers with adjustable complexity levels?

With the labeled dataset from RQ2, we can train a language model to generate answers at different complexity levels based on what the user asks for. There are a few different approaches we could try, like fine-tuning the model on the labeled data, manipulating the model's latent representations, or using reinforcement learning to reward outputs that match the target complexity. The challenge is to find the method that gives the best control over complexity while still producing accurate and relevant answers.

Task 3.1: Fine-tuning Baseline Model

- Fine-tune a pre-trained language model, such as BioBERT or SciBERT, on the complexity-stratified dataset created in RQ2.
- Establish a baseline performance for complexity-controlled answer generation.

Task 3.2: Exploring Advanced Techniques

- Investigate and implement advanced techniques for controllable text generation, such as latent space manipulation, reinforcement learning, or adversarial training.
- Experiment with different approaches and hyperparameters to optimize the model's performance.
- Combine multiple controllable generation techniques to see if they yield cumulative improvements.

Task 3.3: Evaluation and Validation

- Evaluate the performance of the developed models using a combination of automatic metrics and human evaluation.
- Assess the models' ability to control complexity, generate factually accurate and relevant answers, and produce fluent and coherent text.

Task 3.4: (Optional) Integrate external knowledge bases

- Investigate the feasibility of integrating external knowledge bases, such as PubMed or Wikipedia, to improve the model's understanding of medical concepts and terminology.
- Implement a knowledge retrieval mechanism to provide additional context and information for answer generation.
- Evaluate the impact of external knowledge integration on the model's performance and answer quality.

Milestone 3: Development of a controllable language model for generating health answers with adjustable complexity levels.

3.2 Timeline

The following Gantt chart provides a visual representation of the proposed work plan, outlining the tasks and milestones described above, along with other related activities such as writing the final thesis document.

The models will be trained using the Pleiades High Performance Computing (HPC) cluster operated by IEETA. As this is a new setup and I have limited experience with training large language models, the exact timeline may need to be adjusted as the project progresses.

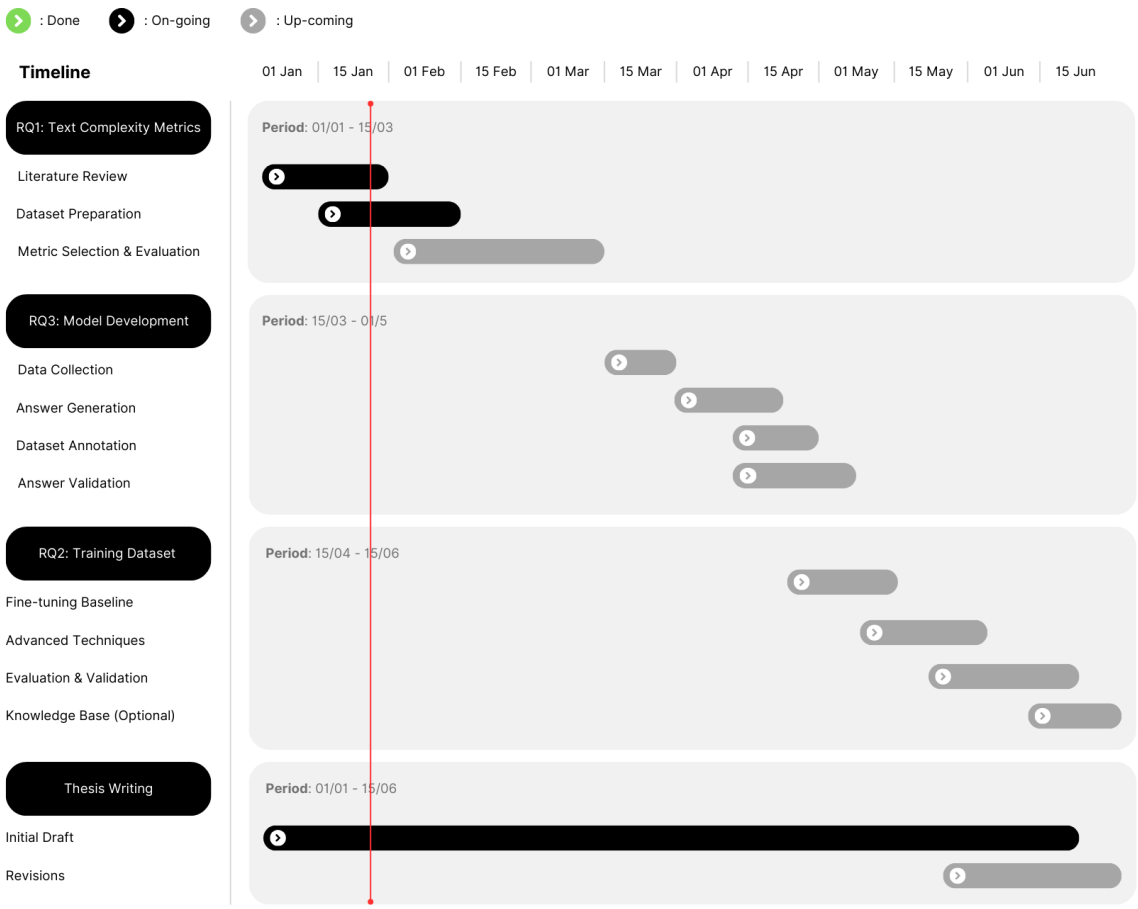

Figure 3.1: Proposed timeline for the thesis project.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].
URL: https://arxiv.org/abs/2005.14165 (cit. on p. 1).

[2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL].
URL: https://arxiv.org/abs/2204.02311 (cit. on p. 1).

[3] Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. *Controllable Text Generation for Large Language Models: A Survey*. 2024. arXiv: 2408.12599 [cs.CL].
URL: https://arxiv.org/abs/2408.12599 (cit. on pp. 4, 6).

[4] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. "A Survey of Controllable Text Generation Using Transformer-based Pre-trained Language Mod-

els." In: *ACM Comput. Surv.* 56.3 (Oct. 2023). ISSN: 0360-0300. DOI: 10.1145/3617680.
URL: https://doi.org/10.1145/3617680 (cit. on p. 4).

[5] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. *CTRL: A Conditional Transformer Language Model for Controllable Generation*. 2019. arXiv: 1909.05858 [cs.CL].
URL: https://arxiv.org/abs/1909.05858 (cit. on pp. 4, 7).

[6] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. *Plug and Play Language Models: A Simple Approach to Controlled Text Generation*. 2020. arXiv: 1912.02164 [cs.CL].
URL: https://arxiv.org/abs/1912.02164 (cit. on pp. 4, 15).

[7] Junxian He, Wojciech Kryscinski, Bryan McCann, Nazneen Rajani, and Caiming Xiong. "CTRLsum: Towards Generic Controllable Text Summarization." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5879–5915. DOI: 10.18653/v1/2022.emnlp-main.396.
URL: https://aclanthology.org/2022.emnlp-main.396/ (cit. on p. 6).

[8] Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. *CoCon: A Self-Supervised Approach for Controlled Text Generation*. 2022. arXiv: 2006.03535 [cs.CL].
URL: https://arxiv.org/abs/2006.03535 (cit. on p. 7).

[9] Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. "POINTER: Constrained Progressive Text Generation via Insertion-based Generative Pre-training." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 8649–8670. DOI: 10.18653/v1/2020.emnlp-main.698.
URL: https://aclanthology.org/2020.emnlp-main.698/ (cit. on p. 8).

[10] Yoel Zeldes, Dan Padnos, Or Sharir, and Barak Peleg. *Technical Report: Auxiliary Tuning and its Application to Conditional Text Generation*. 2020. arXiv: 2006.16823 [cs.CL].
URL: https://arxiv.org/abs/2006.16823 (cit. on p. 9).

[11] Hanqing Zhang and Dawei Song. "DisCup: Discriminator Cooperative Unlikelihood Prompt-tuning for Controllable Text Generation." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 3392–3406. DOI: 10.18653/v

1/2022.emnlp-main.223.
URL: https://aclanthology.org/2022.emnlp-main.223/ (cit. on p. 9).

[12] Chufan Shi, Deng Cai, and Yujiu Yang. *LiFi: Lightweight Controlled Text Generation with Fine-Grained Control Codes*. 2024. arXiv: 2402.06930 [cs.CL].
URL: https://arxiv.org/abs/2402.06930 (cit. on p. 10).

[13] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. *Finetuned Language Models Are Zero-Shot Learners*. 2022. arXiv: 2109.01652 [cs.CL].
URL: https://arxiv.org/abs/2109.01652 (cit. on p. 10).

[14] Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. *Controlled Text Generation with Natural Language Instructions*. 2023. arXiv: 2304.14293 [cs.CL].
URL: https://arxiv.org/abs/2304.14293 (cit. on p. 10).

[15] Vaibhav Kumar, Hana Koorehdavoudi, Masud Moshtaghi, Amita Misra, Ankit Chadha, and Emilio Ferrara. "Controlled Text Generation with Hidden Representation Transformations." In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 9440–9455. DOI: 10.18653/v1/2023.findings-acl.602.
URL: https://aclanthology.org/2023.findings-acl.602/ (cit. on p. 10).

[16] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: *Mach. Learn.* 8.3–4 (May 1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696.
URL: https://doi.org/10.1007/BF00992696 (cit. on p. 11).

[17] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. *Learning to summarize from human feedback*. 2022. arXiv: 2009.01325 [cs.CL].
URL: https://arxiv.org/abs/2009.01325 (cit. on p. 12).

[18] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. *Safe RLHF: Safe Reinforcement Learning from Human Feedback*. 2023. arXiv: 2310.12773 [cs.AI].
URL: https://arxiv.org/abs/2310.12773 (cit. on p. 12).

[19] Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. *A Distributional Approach to Controlled Text Generation*. 2021. arXiv: 2012.11635 [cs.CL].
URL: https://arxiv.org/abs/2012.11635 (cit. on p. 13).

[20] Bhargav Upadhyay, Akhilesh Sudhakar, and Arjun Maheswaran. *Efficient Reinforcement Learning for Unsupervised Controlled Text Generation*. 2022. arXiv: 2204.07696 [cs.CL].
URL: https://arxiv.org/abs/2204.07696 (cit. on p. 13).

[21] Atharva Phatak, David W Savage, Robert Ohle, Jonathan Smith, and Vijay Mago. "Medical Text Simplification Using Reinforcement Learning (TESLEA): Deep Learning–Based Text Simplification Approach." In: *JMIR Med Inform* 10.11 (Nov. 2022), e38095. ISSN: 2291-9694. DOI: 10.2196/38095.
URL: http://www.ncbi.nlm.nih.gov/pubmed/36399375 (cit. on p. 13).

[22] Qingyu Chen, Yifan Peng, and Zhiyong Lu. "BioSentVec: creating sentence embeddings for biomedical texts." In: *2019 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, June 2019. DOI: 10.1109/ichi.2019.8904728.
URL: http://dx.doi.org/10.1109/ICHI.2019.8904728 (cit. on p. 13).

[23] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. "Self-Critical Sequence Training for Image Captioning." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1179–1195. DOI: 10.1109/CVPR.2017.131. (Cit. on p. 14).

[24] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 4222–4235. DOI: 10.18653/v1/2020.emnlp-main.346.
URL: https://aclanthology.org/2020.emnlp-main.346/ (cit. on p. 14).

[25] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. DOI: 10.18653/v1/2021.acl-long.353.
URL: https://aclanthology.org/2021.acl-long.353/ (cit. on p. 15).

[26] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. "P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 61–68. DOI: 10.18653/v1/2022.acl-short.8.
URL: https://aclanthology.org/2022.acl-short.8/ (cit. on p. 15).

[27] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. "GeDi: Generative Discriminator Guided Sequence Generation." In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Punta Cana, Dominican Republic: Associa-

tion for Computational Linguistics, Nov. 2021, pp. 4929–4952. DOI: 10.18653/v1/2021.findings-emnlp.424.
URL: https://aclanthology.org/2021.findings-emnlp.424/ (cit. on p. 15).

[28] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. "DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 6691–6706. DOI: 10.18653/v1/2021.acl-long.522.
URL: https://aclanthology.org/2021.acl-long.522/ (cit. on p. 15).

[29] Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. "Mix and Match: Learning-free Controllable Text Generationusing Energy Language Models." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 401–415. DOI: 10.18653/v1/2022.acl-long.31.
URL: https://aclanthology.org/2022.acl-long.31/ (cit. on p. 15).

[30] Sheng Liu, Haotian Ye, Lei Xing, and James Zou. *In-context Vectors: Making In Context Learning More Effective and Controllable Through Latent Space Steering*. 2024. arXiv: 2311.06668 [cs.LG].
URL: https://arxiv.org/abs/2311.06668 (cit. on p. 16).

[31] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. *Steering Language Models With Activation Engineering*. 2024. arXiv: 2308.10248 [cs.CL].
URL: https://arxiv.org/abs/2308.10248 (cit. on p. 16).

[32] Zhenyi Lu, Wei Wei, Xiaoye Qu, Xian-Ling Mao, Dangyang Chen, and Jixiong Chen. "Miracle: Towards Personalized Dialogue Generation with Latent-Space Multiple Personal Attribute Control." In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5933–5957. DOI: 10.18653/v1/2023.findings-emnlp.395.
URL: https://aclanthology.org/2023.findings-emnlp.395/ (cit. on p. 16).

[33] Adrian C H Fung, Matthew H L Lee, Jessie L Leung, Ivy H Y Chan, and Kenneth K Y Wong. "Internet health resources on nocturnal enuresis: A readability, quality, and accuracy analysis." en. In: *Eur. J. Pediatr. Surg.* 34.1 (Feb. 2024), pp. 84–90. (Cit. on p. 16).

[34] Adam Timothy Lucy, Stephanie L Rakestraw, Courtney Stringer, Daniel Chu, Jayleen Grams, Richard Stahl, and Margaux N Mustian. "Readability of patient education materials for bariatric surgery." en. In: *Surg. Endosc.* 37.8 (Aug. 2023), pp. 6519–6525. (Cit. on p. 16).

[35] Alyssa W Tuan, Nathan Cannon, David Foley, Neha Gupta, Christian Park, Kyra Chester-Paul, Joanna Bhasker, Cara Pearson, Avisha Amarnani, Zachary High, Jennifer Kraschnewski, and Ravi Shah. "Using machine learning to improve the readability of hospital discharge instructions for heart failure." June 2023. (Cit. on p. 16).

[36] Volkan Hanci, Büşra Otlu, and Ali Salih Biyikoğlu. "Assessment of the readability of the online patient education materials of intensive and Critical Care societies." en. In: *Crit. Care Med.* 52.2 (Feb. 2024), e47–e57. (Cit. on p. 16).

[37] Scott Crossley, Aron Heintz, Joon Suh Choi, Jordan Batchelor, Mehrnoush Karimi, and Agnes Malatinszky. "A large-scaled corpus for assessing text readability." In: *Behavior Research Methods* 55.2 (Mar. 2022), pp. 491–507. ISSN: 1554-3528. DOI: 10.3758/s13428-022-01802-x.
URL: http://dx.doi.org/10.3758/s13428-022-01802-x (cit. on p. 17).

[38] Lih-Wern Wang, Michael J. Miller, Michael R. Schmitt, and Frances K. Wen. "Assessing readability formula differences with written health information materials: Application, results, and recommendations." In: *Research in Social and Administrative Pharmacy* 9.5 (2013), pp. 503–516. ISSN: 1551-7411. DOI: https://doi.org/10.1016/j.sapharm.2012.05.009.
URL: https://www.sciencedirect.com/science/article/pii/S1551741112000770 (cit. on p. 17).

[39] Som Singh, Aleena Jamal, and Fawad Qureshi. "Readability Metrics in Patient Education: Where Do We Innovate?" In: *Clinics and Practice* 14.6 (Nov. 2024), pp. 2341–2349. ISSN: 2039-7283. DOI: 10.3390/clinpract14060183.
URL: http://dx.doi.org/10.3390/clinpract14060183 (cit. on p. 17).

[40] Karl Swanson, Shuhan He, Josh Calvano, David Chen, Talar Telvizian, Lawrence Jiang, Paul Chong, Jacob Schwell, Gin Mak, and Jarone Lee. "Biomedical text readability after hypernym substitution with fine-tuned large language models." In: *PLOS Digital Health* 3.4 (Apr. 2024). Ed. by Amara Tariq, e0000489. ISSN: 2767-3170. DOI: 10.1371/journal.pdig.0000489.
URL: http://dx.doi.org/10.1371/journal.pdig.0000489 (cit. on p. 17).

[41] Tsz Ki Ko, Denise Jia Yun Tan, and Ka Siu Fan. "Evaluation of the quality and readability of web-based information regarding foreign bodies of the ear, nose, and throat: Qualitative content analysis." en. In: *JMIR Form. Res.* 8 (Aug. 2024), e55535. (Cit. on p. 17).

[42] Teerapaun Tanprasert and David Kauchak. "Flesch-Kincaid is Not a Text Simplification Evaluation Metric." In: *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*. Ed. by Antoine Bosselut, Esin Durmus, Varun Prashant Gangal, Sebastian Gehrmann, Yacine Jernite, Laura Perez-Beltrachini, Samira Shaikh, and Wei Xu. Online: Association for Computational Linguistics, Aug. 2021, pp. 1–14. DOI: 10.18653/v1/2021.gem-1.1. URL: https://aclanthology.org/2021.gem-1.1 (cit. on p. 17).

[43] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. "Optimizing Statistical Machine Translation for Text Simplification." In: *Transactions of the Association for Computational Linguistics* 4 (2016). Ed. by Lillian Lee, Mark Johnson, and Kristina Toutanova, pp. 401–415. DOI: 10.1162/tacl_a_00107. URL: https://aclanthology.org/Q16-1029 (cit. on p. 17).

[44] Zihao Li, Samuel Belkadi, Nicolo Micheletti, Lifeng Han, Matthew Shardlow, and Goran Nenadic. *Large Language Models for Biomedical Text Simplification: Promising But Not There Yet*. 2024. arXiv: 2408.03871 [cs.CL]. URL: https://arxiv.org/abs/2408.03871 (cit. on p. 18).

[45] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL]. URL: https://arxiv.org/abs/1904.09675 (cit. on p. 18).

[46] Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. "The (Un)Suitability of Automatic Evaluation Metrics for Text Simplification." In: *Computational Linguistics* 47.4 (Dec. 2021), pp. 861–889. DOI: 10.1162/coli_a_00418. URL: https://aclanthology.org/2021.cl-4.28 (cit. on p. 18).

[47] Surendrabikram Thapa, Usman Naseem, and Mehwish Nasim. "From humans to machines: can ChatGPT-like LLMs effectively replace human annotators in NLP tasks." English. In: *Workshop Proceedings of the 17th International AAAI Conference on Web and Social Media*. 17th International AAAI Conference on Web and Social Media, ICWSM 2023 ; Conference date: 05-06-2023 Through 08-06-2023. Association for the Advancement of Artificial Intelligence (AAAI), June 2023. DOI: 10.36190/2023.15. (Cit. on p. 18).

[48] Daniel Deutsch, Rotem Dror, and Dan Roth. "On the Limitations of Reference-Free Evaluations of Generated Text." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 10960–10977. DOI: 10.18653/v1/2022.emnlp-main.753. URL: https://aclanthology.org/2022.emnlp-main.753 (cit. on p. 18).

[49] Elior Sulem, Omri Abend, and Ari Rappoport. "Semantic Structural Evaluation for Text Simplification." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 685–696. DOI: 10.18653/v1/N18-1063.
URL: https://aclanthology.org/N18-1063 (cit. on p. 18).

[50] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. "G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2511–2522. DOI: 10.18653/v1/2023.emnlp-main.153.
URL: https://aclanthology.org/2023.emnlp-main.153/ (cit. on p. 19).