Wrocław University of Science and Technology
Faculty of Information and Communication Technology

Field of study: Sztuczna Inteligencja SZT
Speciality: -

MASTER THESIS

Using Contextual Hallucination Reduction to Improve Answer Reliability in LLMs

Konrad Kiełczyński

Supervisor
dr inż. Jan Kocoń

Keywords:
Large Language Models, Natural Language Processing, Hallucination Reduction, Contextual Hallucination

WROCŁAW (2025)

Contents

Streszczenie

Duże modele językowe stały się podstawowymi narzędziami w przetwarzaniu języka naturalnego, wspomagając zastosowania w szerokim zakresie zadań. Jednak ich powszechne wdrożenie jest utrudnione przez stały problem: *halucynacje kontekstowe*, definiowane jako generowanie wypowiedzi, które są językowo poprawne, lecz niezgodne z faktami zawartymi w kontekście wejściowym. Niniejsza praca podejmuje wyzwanie redukcji halucynacji poprzez wprowadzenie nowych strategii dekodowania, wspomaganych klasyfikatorami opartymi na mechanizmie uwagi. Zastosowano następujące metody: Improved Guided Decoding, Self-Reflection Decoding i Beam+Guided Decoding, które wprowadzają świadomość faktograficzną do procesu generowania bez konieczności modyfikacji architektury modelu bazowego. Obszerne eksperymenty przeprowadzone w ramach wielu rodzin modeli językowych oraz zróżnicowanych zadań wykazały poprawę dokładności. Choć niektóre strategie wiążą się z dodatkowymi kosztami obliczeniowymi wynikającymi z ekstrakcji map uwagi, wzrost niezawodności uzasadnia ten kompromis w dziedzinach wysokiego ryzyka. Podejmując problem halucynacji kontekstowych na etapie inferencji, praca ta przyczynia się do rozwoju bezpieczniejszych i godnych zaufania zastosowań wielkoskalowych modeli generatywnych w przetwarzaniu języka naturalnego.

Abstract

Large Language Models (LLMs) have become foundational tools in Natural Language Processing, powering applications in a wide range of tasks. However, their widespread deployment is hindered by a persistent issue: *contextual hallucination*, defined as the generation of outputs that are fluent but factually inconsistent with the input context. This thesis addresses the challenge of hallucination reduction by introducing novel decoding strategies guided by attention-based classifiers. The following methods have been employed: Improved Guided Decoding, Self-Reflection Decoding, and Beam+Guided Decoding, inject factual awareness directly into the generation process without altering the underlying model architecture. Extensive experiments across multiple LLM families and tasks demonstrate consistent improvements in factual accuracy. While some strategies introduce computational overhead due to attention feature extraction, the gains in reliability justify the trade-off in high-stakes domains. The proposed approaches offer a practical solution to improve answer reliability when retraining is infeasible. By addressing the issue of contextual hallucination at inference time, this work contributes toward safer, more trustworthy applications of large-scale generative models in Natural Language Processing.

1. Introduction

Large language models (LLMs) have gained significant attention in recent years due to their impressive performance on a variety of natural language processing tasks. However, one major challenge they face is the **tendency to hallucinate**. These phenomena occur when models generate content that is either factually incorrect or misaligned with the context they are conditioned on. This thesis focuses specifically on contextual hallucinations, where the generated content differs from the source context (e.g., retrieved documents or user-provided input), even if not obviously false. Hallucinations in LLMs pose significant challenges in real-world applications, including automated content generation, question answering, and summarization, where factual accuracy is paramount. These errors can undermine user trust and the effectiveness of AI-driven systems.

The focus of this work is to propose and evaluate decoding strategies that mitigate contextual hallucinations through classifier-guided methods. These strategies aim to **improve the factual reliability of generated text** without requiring architectural changes or extensive fine-tuning.

1.1 Motivation

This thesis addresses the problem of contextual hallucinations in LLMs, where generated text deviates from the provided input context despite the availability of accurate information. Such hallucinations are particularly problematic in applications that require precise grounding in source material, such as retrieval-augmented generation, domain-specific question answering, and factual summarization.

This work is motivated by the following primary research question:

RQ1: How effectively do hallucination-aware decoding strategies reduce contextual hallucinations compared to standard decoding methods?

RQ2: How well do hallucination-aware decoding strategies generalize across different tasks?

RQ3: What are the computational trade-offs (in GPU memory and decoding latency) of reducing hallucinations using classifier-guided methods?

RQ4: How sensitive is hallucination reduction to the configuration of classifier-guided parameters?

RQ5: How does the nature of the underlying dataset influence the effectiveness of hallucination reduction strategies?

The study is constrained by practical factors such as the availability of computational resources for testing larger LLM models, as well as model, specific limitations, such as certain decoding strategies being incompatible with architectures that use sliding attention. Nevertheless, the proposed methods are designed to be broadly applicable and can be effectively scaled when the necessary infrastructure is available.

1.2 Aims and scope

The primary objective of this thesis is to design and **evaluate classifier-guided decoding strategies to reduce contextual hallucinations during the decoding phase**. To achieve this, research first involves the design and implementation of attention-based classifiers that can detect hallucinations in real time by analyzing internal model states. These classifiers are then integrated into novel decoding strategies that use classifier feedback to influence the token selection process during generation. The effectiveness of these methods is evaluated through comprehensive experiments on benchmark tasks such as question answering and summarization. In addition to performance analysis, the thesis also investigates the trade-offs involved in deploying these techniques, including their impact on accuracy, generalization across tasks and models, and computational efficiency. Furthermore, the study examines how the characteristics of the classifier training data affect the overall performance of hallucination detection and mitigation.

This thesis builds on previous work that I **co-authored**, titled **AggTruth** [20], which proposed attention-based classifiers for detecting contextual hallucinations. Expanding on that foundation, the present work integrates these classifiers directly into the generation process, introducing novel decoding strategies that leverage hallucination detection to improve output factuality.

1.3 Contributions

The contributions of this thesis are as follows:

- Development of multiple hallucination-aware decoding strategies, including *Improved Guided Decoding*, *Self-Reflection Decoding*, and *Beam+Guided Decoding*, aimed at reducing contextual hallucinations in LLM outputs.
- Integration of attention-based hallucination classifiers into the decoding process, enabling context-sensitive token selection through real-time feedback mechanisms.
- Comprehensive evaluation of proposed and widely used decoding methods in various tasks (question answering and summarization) and language model architectures (LLaMA-2, Gemma-2, and Phi-3.5), specifically focusing on their impact on contextual hallucination reduction.

The strengths of this work include its model-agnostic approach, interpretability through attention-based features, and practical relevance to text generation tasks. Limitations include increased computational overhead and the reliance on the quality of classifier training.

1.4 Thesis Outline

This thesis is structured into four main chapters following the introduction. Chapter 2 reviews existing approaches to hallucination detection in LLMs that can later be used to reduce them, covering black-box, white-box, and fine-tuning-based strategies. Chapter 3 outlines the theoretical and technical foundations of the proposed work, including transformer architecture, standard decoding methods, and a set of novel classifier-guided decoding strategies aimed at reducing contextual hallucinations. Chapter 4 presents the experimental setup, detailing the datasets and models used to assess the proposed methods across tasks like question answering and summarization. Finally, Chapter 5 summarizes the findings, highlights practical insights, and proposes directions for future research.

2. Related work

In recent years, the issue of hallucinations in large language models (LLMs) has received considerable interest, with the vast majority of studies focusing on evaluating the factual consistency of the generated text. Within this broader phenomenon, the concept of a **contextual hallucination** has emerged as a distinct subproblem that has increasingly become the subject of specialized investigation. In contrast to general hallucinations, which are characterized by the generation of information that is factually incorrect or entirely fabricated, contextual hallucinations refer to instances where the model's output is inconsistent with the **specific input context on which it was based**, such as a retrieved document or a passage provided by the user. This is a crucial point in applications where LLMs are used for context-sensitive generation, such as retrieval-augmented generation (RAG), grounded dialogue systems, or domain-specific question answering. In such scenarios, it is not enough for an answer to appear entirely convincing; it must also be faithfully anchored in the source content. Contextual hallucinations undermine this requirement and can occur even when the model has access to accurate and relevant information.

The underlying causes of these hallucinations are varied. These include cases where the model misinterprets or gives poor attention to contextual tokens, or where the context itself is characterized by noise, ambiguity, or partial contradictory, resulting in responses that differ from the intended conceptual foundation. In other cases, the hallucination may arise from the model's inadequate attention to critical context tokens during generation, particularly when handling long or complex passages. Although the primary goal of hallucination research is to reduce or eliminate unreliable LLM output, achieving this reliably depends mainly on the ability to accurately detect hallucinations when they occur and mitigate them in the later process. Without robust detection mechanisms, any attempt to reduce hallucinations, whether through training adjustments, architectural changes, or decoding strategies, lacks a foundation even for evaluation and improvement. This concept has led to a variety of strategies for detecting hallucinations, which can be broadly categorized into two methodological paradigms: **black-box** approaches, which analyze model outputs without internal access, and **white-box** methods, which leverage internal representations such as hidden states and attention scores. A third class of techniques, **fine-tuning-based** strategies, aims to reduce hallucinations more directly by adapting the model's behavior through continued training on task-specific datasets.

2.1 Black-Box Methods

Black-box approaches evaluate large language models (LLMs) based only on their outputs, without access to the model's internal states. This makes them especially useful in situations where the model architecture or internal data cannot be accessed, such as with commercial APIs. The output of such detection methods can then be leveraged to guide hallucination reduction strategies, primarily by identifying segments or entire output with a high probability of being hallucinated, which can subsequently be removed or explicitly highlighted to the user.

One of the most influential approaches in this category is **SelfCheckGPT** [19], which assumes that truthful knowledge is reflected in consistent outputs from the same model across multiple independent generations. Specifically, SelfCheckGPT generates multiple output samples for a given input prompt and then evaluates the degree to which these responses support each other. If the outputs reinforce one another by presenting overlapping or mutually consistent information, the content is likely to be reliable. In contrast, a lack of agreement, where one output contradicts or fails to support the others, serves as a strong indicator that the information may be fabricated or ungrounded, i.e., hallucinatory. A key advantage of this technique is that it does not rely on external databases or fact-checking tools, making it model-agnostic. Empirical investigations have demonstrated the effectiveness of this approach in detecting hallucinations at the sentence level in a variety of tasks. However, the method's reliance on multiple forward passes through the model significantly increases computational overhead.

Among recent methods, semantic entropy analysis [9] offers a probabilistic black-box framework for hallucination detection by measuring semantic variability across multiple model outputs. Rather than relying solely on surface-level differences, it quantifies uncertainty in the semantic space, where higher entropy signals a greater risk of confabulation. A distinguishing feature of this approach is its two-stage generation process. After producing an initial response, the model is queried with follow-up factual prompts based on the part of the original output. For example, if the original answer states *Paris is the capital of France*, the model is asked *What is the capital of France?* and similar variants. The semantic entropy of these secondary answers forms the basis for evaluating whether the original statement claim is hallucinated. Although this method achieves high accuracy, its dependence on multiple inference steps makes it computationally demanding, limiting its use in real-time applications.

2.2 White-Box Methods

White-box approaches offer deeper interpretability by analyzing LLM's **internal states**, such as hidden states or attention scores during generation. By revealing how model behaves and transforms at each generation step, they uncover internal patterns that can be leveraged to mitigate hallucinations. These methods are particularly effective in scenarios that require **token level precision** or **real-time hallucination reduction**, where timely and fine-grained insights into model decision making are critical.

The fundamental white-box method is **hidden state** analysis [2], the authors used these internal states from different layers of large language models (LLMs) to investigate how factuality is encoded in these representations during model generation. By training binary classifiers on these features, Azaria and Mitchell demonstrated that hidden states within transformer architecture carry latent, interpretable features that correlate with the factual accuracy of the model's outputs. This approach was the first to highlight the potential of internal model states as a source of supervision for factuality detection, offering a transparent alternative to post hoc evaluations. As a baseline method, hidden state analysis establishes an important link between internal representations and high-level semantic properties, anchoring future work in white-box interpretability.

Another method, **Lookback Lens** [6], demonstrates that attention scores can also serve as effective features for both detecting and reducing hallucinations. The core of this approach is the introduction of the *lookback ratio*, a metric (attention aggregation) that measures how much attention a language model assigns to the prompt and context compared to its own generated content during text generation. For each attention head across all layers, a single ratio is computed, resulting in feature vectors where each feature corresponds to one attention head. These vectors are then used to train a simple logistic regression model, which predicts whether a segment of generated text (defined as a sliding window of eight tokens) is likely to be factual or hallucinated. The method has shown robust performance across various tasks, including summarization, question answering, and cross-task transfer scenarios. Furthermore, the authors applied the trained classifier in the **decoding process** to guide generation decisions, effectively reducing hallucination rates. This **integration of detection into generation** itself serves as a central inspiration for the methods proposed in this thesis.

Expanding on this attention-centric paradigm, the **AggTruth [20] paper, co-authored by myself and colleagues from university**, introduces a comprehensive framework for contextual hallucination detection based on the aggregation of attention scores. The core contribution of AggTruth is a suite of aggregation strategies; Sum, Cosine Similarity, Entropy, and Jensen-Shannon Divergence (JS-Div); which quantify the attention paid to the passage or context and extract fine-grained features for hallucination detection. These aggregated features are computed over sliding windows of eight generated tokens, enabling localized token-level detection of hallucinations during generation. To further enhance computational efficiency, performance, and interpretability, AggTruth incorporates feature selection methods such as Spearman correlation and Lasso regularization to identify the most informative attention heads. Empirical evaluations show that AggTruth outperforms both hidden-state-based classifiers and prior attention-based baselines, including Lookback Lens, across a range of tasks such as

question answering, summarization, and most notably cross-domain generalization. This strong performance indicates that the attention-based features leveraged by AggTruth capture task-agnostic patterns, allowing the model to generalize effectively across different input formats and domains. This is particularly valuable in real-world applications, where models often face inputs from unseen or heterogeneous sources. Its lightweight architecture and its reliance on internal attention scores make it particularly suitable for **white-box real-time hallucination detection and reduction**, demonstrating its suitability to build more reliable and context-based LLM systems. The AggTruth framework is adopted as the **foundation detection component of the hallucination reduction method proposed in this thesis**. Its ability to extract fine-grained interpretable features from attention scores,combined with demonstrated robustness across tasks, makes it an ideal candidate for integration into a real-time reduction system that modifies the generation strategy of large language models.

2.3 Complementary Strategy: Fine-Tuning

Although hallucination detection forms the foundation for any effective reduction strategy, some approaches seek to mitigate hallucinations during the training phase of the model through fine-tuning. This strategy involves updating LLMs on **domain-specific** or factuality-annotated datasets, with the goal of reinforcing truthful generation.

However, recent research has highlighted several important limitations of this strategy. [10] suggest that fine-tuning LLMs with new factual information, particularly content not seen during pretraining, can unintentionally increase the frequency of hallucinations. Their controlled experiments reveal that while models eventually learn this new information, it is acquired more slowly and correlates with a higher tendency to generate factually incorrect answers. Moreover, fine-tuning poses the well-documented risk of **catastrophic forgetting**, where previously acquired general knowledge is reduced as a result of overfitting to a specific domain. Studies by [14, 18] confirm that continued training on limited datasets can significantly degrade generalization performance and the model's ability to preserve tasks mastered during pretraining. Therefore, while fine-tuning holds potential as a hallucination mitigation tool, it must be applied judiciously and potentially with robust detection/reduction mechanisms. Without detection-guided optimization or evaluation, fine-tuning alone may introduce new failure modes and undermine the reliability it seeks to improve.

2.4 Summary

In summary, existing approaches to hallucination detection and reduction in LLMs have evolved from surface-level output analysis to fine-grained inspection of internal model states. Black-box methods such as SelfCheckGPT [19] and semantic entropy analysis [9] offer accessibility and model-agnostic deployment, but suffer from high computational costs and limited real-time applicability. In contrast, white-box methods provide deeper interpretability by leveraging hidden states and attention scores, as exemplified by Lookback Lens [6] and AggTruth [20]. These techniques enable localized and efficient hallucination detection, with AggTruth, in particular, offering a robust and lightweight framework suitable for real-time integration. Although fine-tuning [10, 14, 18] offers another axis of control, its use is constrained by domain-specific risks and generalization challenges such as catastrophic forgetting and overfitting.

Together, these methods form a rich methodological landscape, but few have explored the direct integration of detection into the decoding process. This thesis addresses that gap by incorporating white-box classifiers into novel decoding strategies, enabling generation that is not only fluent but also contextually faithful.

3. Methods

In the field of natural language processing, transformer-based architectures [33] have emerged as the dominant backbone for state-of-the-art language models. This chapter presents the foundational components and processes of these architectures, with a particular emphasis on **decoder-only models**[26], which form the basis of modern large language models (LLMs). These models operate in an **autoregressive** manner, processing input text sequentially, and predicting the next token conditioned on all previously generated tokens. Their structure, training paradigm, and decoding strategies significantly influence both their performance and their vulnerability to **producing hallucinated content**.

To understand and ultimately mitigate such phenomena, it is essential to examine how these models interpret and generate text. In Section 3.1, we outline the **input processing and encoding** pipeline, describing how raw text is transformed into a structured representation through tokenization (Section 3.1.1), embedding (Section 3.1.2) and positional encoding (Section 3.1.3). Section 3.2 introduces the core components of the **transformer architecture**, including multi-head self-attention (Section 3.2.2), and highlights their roles in representation learning and token generation. Next, Section 3.3 explores various **decoding strategies** employed during inference, such as greedy decoding, beam search, or DoLa [7]. Finally, Section 3.4 introduces the **hallucination reduction method** incorporated into the decoding process. Together, these sections provide a structured basis for evaluating hallucination reduction strategies and accuracy in decoder-only transformer-based language models.

3.1 Input Processing and Encoding

3.1.1 Tokenization

Tokenization is the process of breaking down input text into smaller, more manageable units known as tokens. These tokens serve as the fundamental interface between raw textual input and the numerical representations consumed by language models. Traditional approaches to tokenization often relied on whitespace or punctuation-based segmentation, which proved insufficient for morphologically rich languages and prone to high rates of out-of-vocabulary (OOV) [21].

As a precursor to modern subword tokenization, **FastText** [4] introduced an efficient use of character-level n-grams, representing each word as a bag of substrings. This technique enabled generalization over rare or unseen words, particularly beneficial in low-resource or morphologically complex settings.

Building on these ideas, modern LLMs typically rely on subword tokenization methods such as **Byte Pair Encoding** (BPE) and **WordPiece** [29, 36]. These algorithms decompose words into frequent subword units, effectively reducing vocabulary size while maintaining coverage of diverse linguistic forms. BPE iteratively merges the most frequent pair of adjacent symbols (characters or subwords) in the corpus, while WordPiece selects subword units based on maximizing the likelihood of the training data under a language model. A related technique, **Unigram Language Model** tokenization [16], selects an optimal subset of tokens from a larger vocabulary using a probabilistic approach. Although conceptually distinct, these methods share a reliance on subword information to mitigate the OOV problem. In the context of LLMs, tokenization is not a trivial preprocessing step, it fundamentally impacts model performance by determining input granularity, influencing attention patterns, and shaping the contextual understanding and reasoning abilities of the model.

3.1.2 Word Embeddings

In natural language processing, word embeddings are dense vector representations that encode semantic relationships based on patterns of word co-occurrence in large text corpora. Foundational techniques such as Word2Vec [21], GloVe [24], and FastText [4] implement this concept by mapping semantically similar words to nearby points in the embedding space. A key feature of these embeddings is their capacity to perform vector **arithmetic-based analogies** (e.g., $\texttt{king} - \texttt{man} + \texttt{woman} \approx \texttt{queen}$), which reveal linear structures that encode semantic relations.

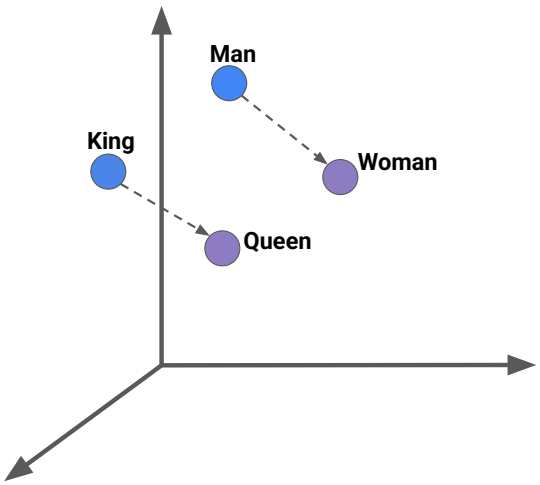


Figure 3.1: Visualization of an embedding space capturing semantic relationships, such as the analogy $\texttt{king} - \texttt{man} + \texttt{woman} \approx \texttt{queen}$. While this example stems from word embeddings, similar principles extend to token embeddings used in LLMs.

These early methods produce static embeddings, meaning that each word has a fixed

vector regardless of the context in which it appears. Although computationally efficient, this approach fails to account for polysemy, the phenomenon where a word has multiple meanings depending on context.

With the advent of transformer-based large language models (LLMs), the notion of embeddings evolved into contextualized token embeddings. Unlike static word embeddings, token embeddings are dynamically generated for each occurrence of a word, allowing the model to disambiguate meaning based on context. These embeddings operate at the subword level, using tokenization schemes such as Byte Pair Encoding or WordPiece (Section 3.1.1), and are learned jointly with model training, enabling fine-grained representation of both common and rare linguistic units. Although Figure 3.1 is traditionally associated with static word embeddings, the semantic structures it illustrates remain relevant to contextualized token embeddings in large language models, particularly when token units correspond to entire words.

3.1.3 Positional Encoding

Transformers process tokens in parallel, operating on all input elements simultaneously rather than sequentially. As a result, this architecture lacks an inherent sense of token order, since the self-attention mechanism (Section 3.2.2) treats inputs as position agnostic. Without positional information, a model would process the sentences *dog chases ball* and *ball chases dog* identically, despite their clearly different meanings. This position awareness is critical for interpreting grammatical structures and semantic dependencies.

To address this, positional encodings are used, vectors that are added to token embeddings before any attention computations. These encodings inject information about the position of each token in the sequence. They may be derived from fixed mathematical functions, such as sinusoidal functions introduced in the original Transformer architecture [33], or learned during training as part of the model parameters. Fixed encodings provide an inductive bias toward regular patterns, while learned encodings can adapt to the specific characteristics of the training data. In both cases, they allow the model to distinguish between tokens based not just on content but also on order.

In large language models (LLMs), the role of positional encoding becomes even more critical, especially for handling long-range dependencies and maintaining alignment between generated tokens and the structure of the input prompt. Poor handling of positional information can lead to coherence breakdowns, hallucinations, or structural inconsistencies in generated outputs, particularly as context lengths grow.

3.2 Transformer Architecture

Before the advent of transformers, sequential models like Recurrent Neural Networks (RNNs) [27] and Long Short-Term Memory (LSTM) networks [13] were the predominant architectures for natural language processing tasks. These models process input tokens strictly one at a time, updating a hidden state sequentially. This inherently sequential nature limited their efficiency in training, as computations could not be parallelized across tokens. Moreover, modeling long-range dependencies was challenging due to vanishing gradients and memory bottlenecks.

Transformers fundamentally changed this paradigm through the introduction of the self-attention mechanism, as described in Section 3.2.2. Unlike RNNs and LSTMs, transformers do not rely on past hidden states for future computations, allowing them to process entire input sequences in parallel. This architectural innovation leads to significantly **faster training** and enables scaling to much **larger datasets**. Furthermore, self-attention allows each token to dynamically attend to all others in the sequence, effectively capturing long-range dependencies that were previously difficult to model. These advantages have established transformers as the backbone of virtually all state-of-the-art natural language processing (NLP) models [33, 15, 38].

3.2.1 Model Overview

Large Language Models (LLMs), such as those used in this study, predominantly leverage the transformer architecture. This architecture, originally proposed by Vaswani et al. [33], has become the foundation for most state-of-the-art language generation systems due to its scalability and effectiveness in capturing long-range dependencies.

While the original transformer design consists of both encoder and decoder stacks, many modern LLMs, such as the LLaMA [32] and GPT variants [5, 23], adopt a **decoder-only** configuration. This choice is particularly well-suited to autoregressive generation tasks, in which a model produces one token at a time, conditioning on all previously generated tokens. The decoder-only transformer simplifies training and inference, while maintaining high performance across a wide range of NLP tasks. These models are **typically trained on massive corpora** using unsupervised objectives such as causal language modeling, enabling them to learn rich representations of language and world knowledge. A visual representation of the decoder-only architecture is shown in Section 3.2.

The following subsections provide a detailed breakdown of its key components: multi-head self-attention (Section 3.2.2), residual connections with layer normalization (Add & Norm) (Section 3.2.3), feed-forward networks (Section 3.2.4), and the generation of output logits (Section 3.2.6).

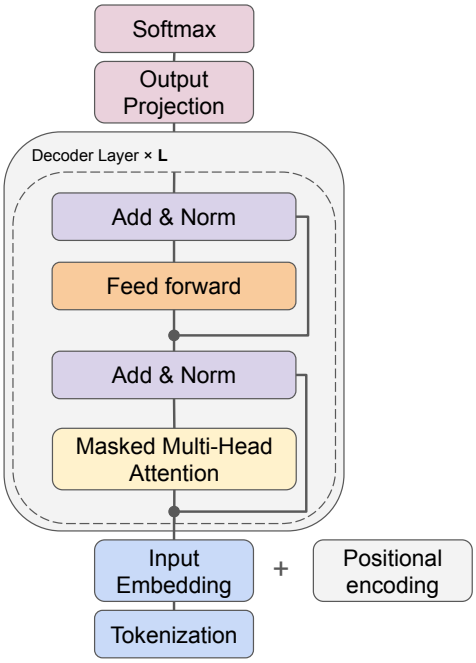
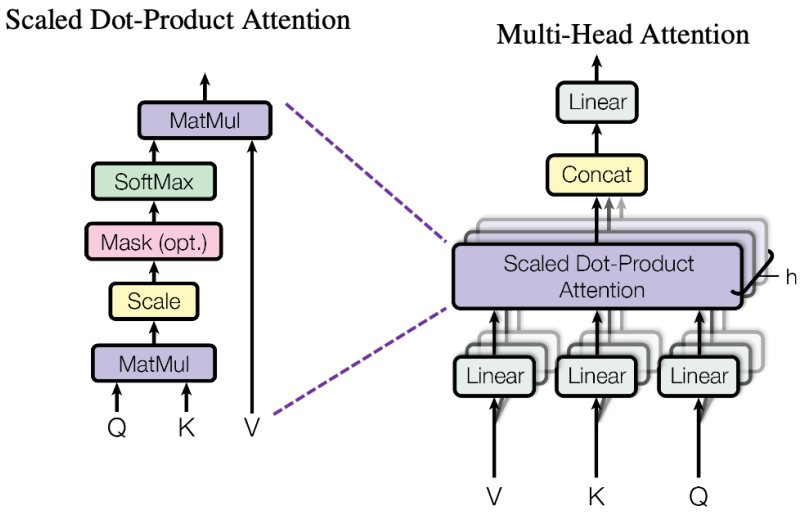


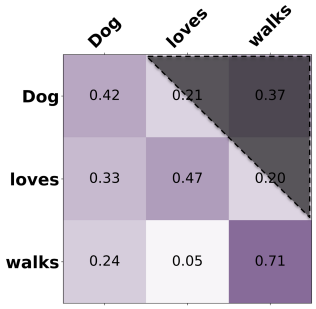
Figure 3.2: A visual representation of the decoder-only Transformer architecture, commonly used in autoregressive large language models (LLMs). The model begins by processing input tokens through tokenization and input embeddings, with positional encodings added to retain order information. It then passes through a stack of L identical decoder layers, each comprising masked multi-head self-attention, a feed-forward network, and residual connections followed by layer normalization.

3.2.2 Multi-Head Self-Attention

The foundational mechanism behind modern transformer architectures is the Multi-Head Self-Attention mechanism, first introduced in the seminal work **"Attention is All You Need"** by Vaswani et al. [33]. At its core, attention is a mechanism for computing relevance between elements within a set (e.g., words/tokens), allowing a model to dynamically weigh the importance of input components in relation to others. This makes attention fundamentally about information relevance rather than sequence modeling. Notably, attention mechanisms have been employed well beyond transformers. Importantly, attention is a general-purpose computation strategy that has been successfully adopted in a wide variety of domains and architectures beyond the original transformer framework. For instance, Graves (2014) [11] demonstrated the use of differentiable attention in neural Turing machines, enabling models to focus selectively on memory locations, a precursor to modern attention concepts. In computer vision, models like the Vision Transformer (ViT)[8] and CLIP[25] leverage attention to capture spatial and semantic relationships within and across modalities. Moreover, attention mechanisms have also proven effective in non-transformer architectures. Furthermore, Graph Attention Networks (GATs)[34] extend attention to graph-structured data, allowing adaptive weighting of neighboring nodes. Similarly, convolutional block attention modules (CBAM)[35] enhance convolutional neural networks by applying attention across channel dimensions. This widespread applicability underscores the versatility and power of attention as a computational mechanism.



(a) Original multi-head attention architecture [33].



(b) Softmax-normalized self-attention scores with a causal mask applied.

Figure 3.3: Visualization of multi-head attention and softmax-normalized attention scores, optionally with a causal mask applied (visible as suppression along the upper triangle).

After introducing the foundational importance of attention, it is essential to dive into the mechanism itself. In the self-attention mechanism, each input token is first projected into three distinct vectors: query Q, key K, and value V. These vectors are computed as:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where X is the input matrix and W^Q, W^K, W^V are learned projection matrices. Conceptually, we can understand these components by their roles: a query represents what a token *is looking for* from other parts of the sequence. A key represents what a token *offers* or *reveals about itself* or reveals about itself to others. The value contains the *actual information* a token carries and is shared with other tokens if their queries are well aligned with its key.

The attention output is calculated using the scaled dot-product attention formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where d_k is the dimensionality of the key vectors and acts as a scaling factor to mitigate gradient instability. The softmax function transforms the raw attention scores into a probability distribution 3.3b. During generation tasks, a causal mask is typically applied before the softmax to prevent the model from attending to future tokens, enforcing the autoregressive property.

Multi-head attention shown in Figure 3.3a, extends this mechanism by performing the attention computation in parallel across h independent attention heads:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

Each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

This setup allows each head to attend to different parts of the input sequence and learn different relational patterns. By aggregating information from multiple representational subspaces, the model captures granular dependencies and interactions between tokens. After concatenating the outputs of all attention heads, a learnable projection matrix W^O is applied to transform the result back into the hidden dimension of the model, allowing seamless integration with subsequent layers. This architectural feature is a key component of large language models (LLMs), allowing them to function as **deeply contextual models**, capable of dynamically adapting their attention based on the surrounding context, thereby improving the coherence, factual alignment, and semantic depth of the generated output.

3.2.3 Residual Connection and Layer Normalization

Residual connections, originally introduced in ResNet architectures [12], are a key component in transformer models, including large language models (LLMs). In the transformer architecture, each sublayer, such as multi-head attention or feed-forward layers, is wrapped with a residual connection, where the output of the sublayer is added element-wise to its input before passing through layer normalization. Mathematically, given an input x and a sublayer function $\text{Sublayer}(x)$, the output is computed as $\text{LayerNorm}(x + \text{Sublayer}(x))$. This design is crucial for maintaining stable training dynamics in deep networks, as it allows gradients to propagate more effectively through many layers, mitigating the problem of vanishing gradients. In addition, residual connections preserve the original input signal, allowing the model to learn incremental transformations rather than entirely new representations at each layer. Layer normalization, in turn, helps normalize the inputs to each sublayer, reducing internal covariate shift and improving convergence speed and generalization.

3.2.4 Feed-Forward Network

The Feed-Forward Network (FFN) is part of the transformer architecture, placed after the attention sublayer within each transformer block. Typically consisting of two linear transformations with a nonlinear activation function (such as ReLU or GELU) in between, the FFN enables the model to capture complex, nonlinear relationships and enrich token representations. This transformation allows the model to encode richer and more abstract representations of the input, going beyond what attention alone can achieve. The intermediate layer in the FFN often has a significantly higher dimensionality than the model dimension. This expansion, followed by a nonlinear transformation, enhances the model's ability to model complex patterns and interactions in the data.

3.2.5 Hidden States

In transformer-based large language models (LLMs), hidden states are internal vector representations that encode the evolving **contextual meaning of input tokens** as they traverse successive layers of the model. Each transformer layer computes a new hidden state for every token, capturing both the token's identity and its interactions with other tokens in the sequence. These interactions are mediated by mechanisms such as self-attention and layer normalization, which allow each token to attend to relevant parts of the input. As tokens progress through the layers, their hidden states are progressively refined, integrating richer syntactic and semantic dependencies. Early layers often capture surface-level features (e.g., part-of-speech or syntax), while deeper layers abstract higher-level concepts such as entity relations or discourse structure. In the final layer, hidden states represent a deeply contextualized understanding of each token, combining both the local and global context.

These representations are fundamental to the model's ability to perform downstream tasks such as text classification, text generation, and question answering. Moreover, hidden states are frequently used for model interpretability, probing analyses, and

transfer learning in applications such as sentence embeddings or fine-tuning for specific domains.

3.2.6 Output Logits

Output logits are the raw, unnormalized scores generated by applying the final linear transformation to the model's last hidden states (Section 3.2.5). Each logit represents the model's internal assessment of a vocabulary token's suitability as the next output. These scores encapsulate both the contextual information embedded in the hidden states and the learned mapping from internal representations to specific vocabulary items. Typically, these logits are passed through a softmax function to generate a probability distribution across the entire vocabulary. This distribution serves as the foundation for text generation strategies, as described in Section 3.3.

3.3 Standard Text Generation and Decoding Strategies

Once the logits are obtained, they are used in text generation. The decoding strategies determine how these raw scores are transformed into human-readable text. The strategies balance between diversity and determinism in the output. Different decoding methods present distinct trade-offs, and each one is suited for specific tasks depending on whether more creativity or consistency is desired in the generated text.

3.3.1 Greedy Search

Greedy search is one of the simplest decoding strategies. It generates text by selecting the most likely token at each step based on the model's output probabilities. Given a sequence of logits, the greedy approach computes the next token by choosing the one with the highest probability:

$$y_t = \arg\max P(y_t \mid y_1, y_2, \ldots, y_{t-1})$$

where y_t is the token predicted in time step t, and the probability $P(y_t \mid y_1, y_2, \ldots, y_{t-1})$ is derived from the softmax transformation of the logits at each step. This strategy ensures that each token is the most probable, **leading to repetitive deterministic text**.

3.3.2 Sampling

Sampling introduces an element of randomness to the generation process, which increases the diversity of the output. In this method, the model generates text by sampling from the probability distribution of possible next tokens, rather than choosing the most probable one. The next token y_t is sampled from the probability distribution $P(y_t \mid y_1, y_2, \ldots, y_{t-1})$.

Temperature

A key parameter in sampling is the **temperature** (T), which controls the randomness of the sampling process. It is applied to the logits before the softmax operation, altering the probability distribution:

$$P(y_t = i \mid y_1, y_2, \ldots, y_{t-1}) = \frac{e^{l_i/T}}{\sum_j e^{l_j/T}}$$

where (l_i) are the logits corresponding to each token (i), and the sum in the denominator normalizes the distribution. The temperature parameter has the following effects:

- Low temperature (e.g., $T < 1$): Sharpens the distribution, making it more deterministic and favoring high-probability tokens.
- High temperature (e.g., $T > 1$): Flattens the distribution, encouraging more exploration and diversity at the cost of coherence.
- Temperature ($T = 0$): Equivalent to greedy search, where the token with the highest probability is always selected.

Temperature allows precise control over the trade-off between creativity and consistency in the generated text.

Top-k

Top-k sampling introduces another way to control randomness in text generation by limiting the number of tokens from which the model can sample at each step. Instead of sampling from the full distribution of possible next tokens, the model only considers the top k most probable tokens. The probability distribution is renormalized on these k tokens, and then the next token is sampled from this restricted set. This approach ensures that less probable tokens are excluded, preventing the model from generating non-sensical or irrelevant outputs.

Top-p

Top-p sampling, also known as **nucleus** sampling, is another method that introduces randomness to the text generation process while focusing on a dynamic set of possible tokens. Instead of selecting a fixed number of top tokens (k), top-p sampling selects the smallest set of tokens whose cumulative probability is greater than or equal to a threshold p. This set is referred to as the *nucleus*.

Mathematically, the model first sorts the tokens by their probability and selects the smallest subset of tokens such that:

$$\sum_{i \in \text{top-p}} P(y_i \mid y_1, y_2, \ldots, y_{t-1}) \geq p$$

where p is a parameter between 0 and 1. The next token is then sampled from this subset. This ensures that the probability mass is concentrated in the most likely tokens, but the size of the set of possible tokens varies depending on the shape of the distribution.

3.3.3 Beam Search

Beam search is a heuristic search algorithm commonly used in sequence generation tasks. Unlike greedy search, which selects the highest-probability token at each step, beam search maintains a fixed number of top candidate sequences, known as the `beam width/size`, across each decoding step. This approach enables the model to explore a more diverse set of candidate sequences, potentially leading to more coherent outputs.

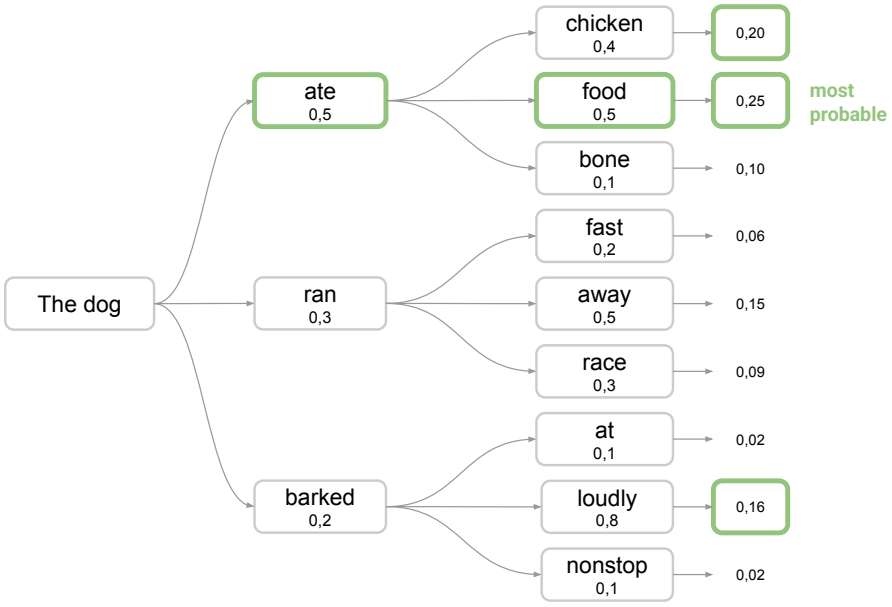


Figure 3.4: Visualization of the beam search algorithm with a beam width of 3. At each step, the algorithm selects the top 3 most probable next words, highlighted in green. The most probable path is determined at the end by the highest cumulative probability. For simplicity, raw probabilities are used for illustration; in practice, log-probabilities are used.

In practice, beam search operates on log-probabilities to prevent numerical underflow and to simplify the computation of cumulative scores, as the sum of log-probabilities rather than the product of raw probabilities, and for improved numerical stability. However, for clarity and ease of visualization, the accompanying figure and the example use raw probabilities.

This improved exploration comes at a computational cost: as the beam width increases, the number of sequences evaluated and stored at each step also grows. Therefore, the choice of beam width represents a trade-off between output quality and computa-

tional efficiency, an important consideration in real-time or resource-constrained inference settings.

Figure 3.4 illustrates a beam search process with a beam size of 3. Starting from the initial token, *The dog*, the algorithm explores multiple branching paths corresponding to candidate tokens such as *ate*, *ran*, and *barked*, each associated with a conditional probability. At each step, the algorithm retains the top three most probable sequences, highlighted in green. Cumulative probabilities, shown here as products of raw values, are used to classify the paths. For instance, the sequence *The dog ate food* has the highest cumulative probability ($0.5 \cdot 0.5 = 0.25$), making it the most likely output. This process demonstrates how beam search balances local token likelihoods with global sequence coherence, often outperforming greedy decoding by **avoiding early commitment to suboptimal tokens**.

3.3.4 DoLa

DoLa [7] (Decoding by Contrasting Layers) is a decoding strategy designed to reduce hallucinations in large language models (LLMs) by leveraging the hierarchical encoding of factual knowledge across transformer layers. Unlike conventional approaches that depend solely on the output of the final layer, DoLa compares the logits of the earlier and later layers to better extract factual content embedded throughout the model. The underlying hypothesis is that factual tokens exhibit a consistent increase in activation scores as information propagates through the network, indicating growing model confidence (see Figure 3.5). In contrast, spurious or inaccurate tokens tend to display flat or declining trajectories.

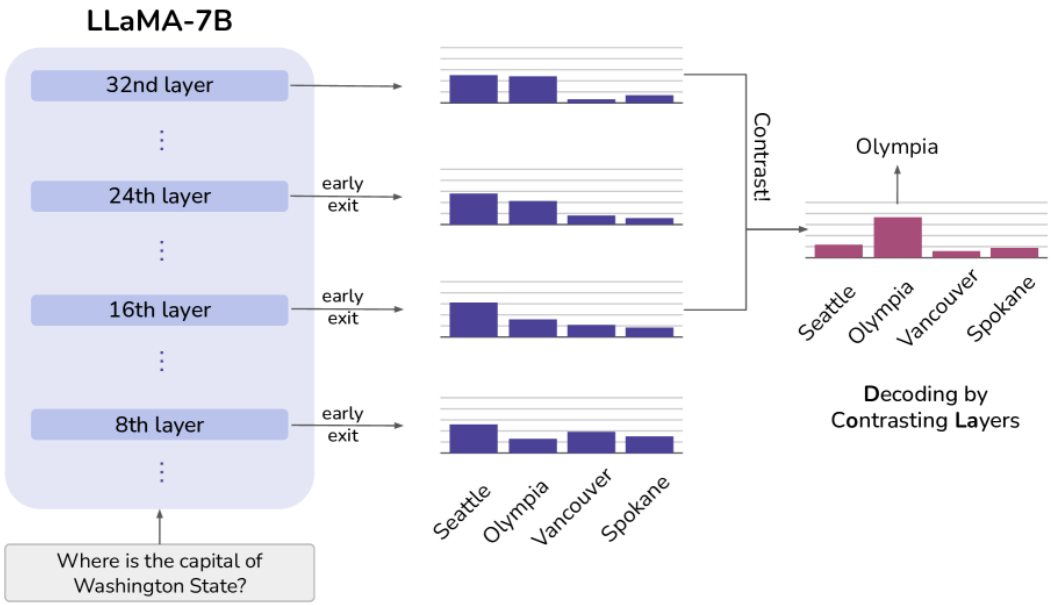


Figure 3.5: Visualization of DoLa from [7]. As the language model processes the input question through deeper layers, the probabilities for factual tokens such as *Olympia* increase, while alternatives such as *Seattle* remain relatively stable.

Method follows the mechanism of early exit [31], not by terminating inference prematurely, but by applying language modeling heads to intermediate hidden states to inform token selection during decoding. The selection of layers used for contrast is governed by the `dola_layers` parameter, which can be specified directly (via layer indices) or through short-hand values such as `low` or `high`, denoting lower or higher transformer layers, respectively. For tasks like short-answer fact verification, the `high` setting is typically more effective, reflecting the sharper confidence signal of the upper layers. In contrast, tasks that require multistep reasoning or long-form generation may benefit from the `low` setting, as reasoning signals often emerge earlier in the computation. By prioritizing tokens with upward logit trends, DoLa steers generation toward more factual outputs, all without requiring external knowledge bases or additional fine-tuning, making it both efficient and model-agnostic.

3.4 Classifier-Guided Decoding Methods

An effective approach to improving the factual quality of generated text involves guiding the decoding process using a dedicated classifier. In this paradigm, a separate model is trained to assess the reliability or contextual consistency of generated sequences, and its predictions are used to influence generation decisions. By tailoring the classifier to specific tasks or domains, it can provide targeted feedback that enhances the trustworthiness of the output. The following subsections describe **several decoding strategies**, some developed in prior work, others **proposed in this thesis**, that leverage such classifier-based guidance.

These methods are designed as direct responses to the research problems outlined in Section 4.1. In particular, they aim to improve the factual consistency of generated text (RQ1), test generalization across tasks and models (RQ2), assess computational trade-offs such as latency and memory usage (RQ3), evaluate sensitivity to decoding parameters (RQ4), and analyze the impact of dataset characteristics (RQ5). These methods enable real-time, context-sensitive control over token generation by integrating hallucination detection directly into the decoding process, primarily through attention-based classifiers

3.4.1 Lookback Lens Guided Decoding

To mitigate the impact of contextual hallucinations identified by the Lookback Lens classifier [6], a classifier-guided decoding strategy is introduced to steer generation toward contextually grounded outputs. Unlike prior approaches that control text generation by adjusting output probabilities based on token-level classifier scores, this method fundamentally differs by leveraging attention maps during generation instead of the tokens themselves.

In this approach, the Lookback Lens classifier is integrated into the decoding process to evaluate and select among candidate continuations, each comprising a multi-token chunk (typically 8 tokens). At each decoding step, multiple candidate chunks are independently sampled on the basis of the current context and the partially generated

output. For each candidate, the features derived from the attention patterns are averaged to represent their retrospective influence on the prior context. The classifier then assesses these representations and selects the candidate that best aligns with the preceding content and is most truthful. Once selected, the candidate is appended to the generated sequence and the process repeats until an end-of-sequence token is produced or a maximum sequence length is reached. This multi-chunk evaluation enables the method to maintain global contextual alignment, thereby improving coherence and reducing hallucinations, all without requiring model fine-tuning or external knowledge sources. However, the method generates candidate chunks sequentially and does not utilize caching, resulting in significantly slower performance compared to more efficient decoding strategies.

3.4.2 Improved Guided Decoding ***Proposed***

Building upon the Lookback Lens Guided Decoding approach, I propose an enhanced guided decoding method that incorporates classifier-based selection while significantly improving computational efficiency. The method addresses the latency of the original Lookback Lens strategy by leveraging caching mechanisms and batched candidate generation. Firstly, the key-value pairs associated with the prompt backbone are cached prior to generation, enabling their reuse across decoding steps. This avoids redundant computation and accelerates the generation process. Subsequently, k candidate continuations are generated in parallel through batched inference, allowing the model to produce multiple candidate chunks simultaneously rather than sequentially.

The classifier then evaluates each candidate and the one with the highest score is selected as the next continuation. This batched evaluation ensures that the selected output aligns well with the prior context and reduces hallucinations by integrating classifier-guided constraints. By combining efficient batched generation with classifier-based selection, this method achieves a favorable balance between decoding speed and output quality. In contrast to the original Lookback Lens decoding, which performs sequential candidate generation and lacks caching, this approach significantly improves runtime efficiency while preserving contextual fidelity.

3.4.3 Self-reflection Decoding ***Proposed***

Self-reflection decoding is another generation method designed to reduce hallucinations by incorporating the model's internal confidence into the decoding process. The method allows the model to reflect on its own predictions and dynamically review them.

Initially, the model generates tokens using a standard strategy such as greedy decoding. After generating a token t_k, the model continues with t_{k+1} and, based on this continuation, evaluates the likelihood that t_k was a hallucination. This evaluation is typically based on internal states such as attention, which become computed only after the next token is produced. If a high probability of hallucination (threshold exceeded) is detected for t_k, the self-reflection mechanism is activated. The model returns to the previous token t_{k-1} and considers the top-k alternative candidates for t_k. For each candidate, the model generates a potential t_{k+1} and computes the probability of hallucination p_H. Among these, the version of t_k that results in the lowest p_H is

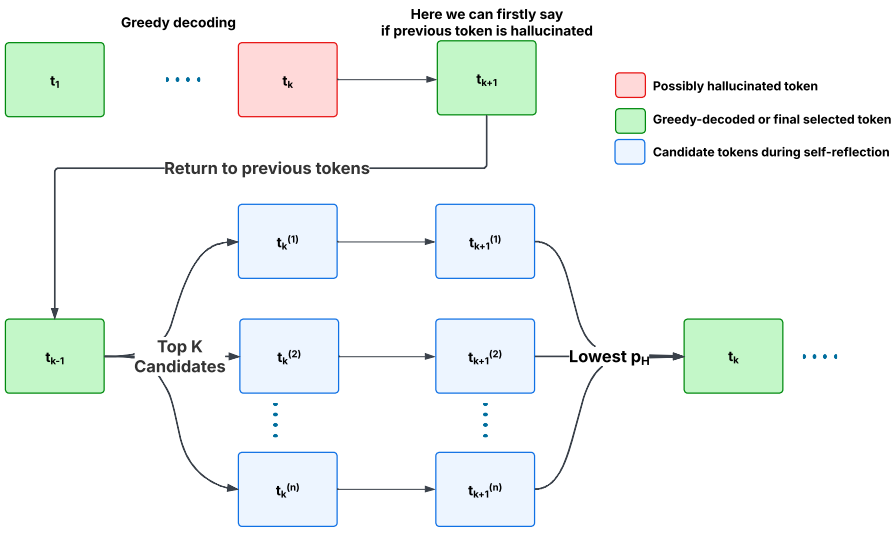


Figure 3.6: Overview of the Self-reflection Decoding pipeline. Initially, greedy decoding proceeds until token t_k. After generating t_{k+1}, the model assesses whether t_k might be hallucinated. If so, the decoding returns to t_{k-1} and generates top-K candidate tokens for t_k. Each candidate is followed by a new t_{k+1}, and the candidate with the lowest hallucination probability p_H is selected.

selected and the generation process continues from there. This pipeline is illustrated in Figure 3.6. The upper section shows the initial greedy decoding process, while the lower section details the self-reflection mechanism where alternative candidates are evaluated and selected.

3.4.4 Beam+Guided ***Proposed***

Finally, I propose a novel decoding strategy called *Beam+Guided Decoding*, which merges the efficiency of traditional beam search with the factual accuracy encouraged by classifier-guided decoding. The core idea is to preserve the structure of the standard beam search while improving it with a classifier-derived *truthfulness score*, encouraging the model to generate sequences that are not only likely, but also factually reliable.

In standard beam search (see Figure 3.4), the top-k partial sequences are selected at each step based on their cumulative log-probabilities. Beam+Guided Decoding improves this by re-scoring each hypothesis $y^i_{1:t}$ through a linear interpolation of two components: (1) the cumulative log-probability of the sequence under the model up to token t, and (2) the cumulative log-probability from a classifier assessing the factual consistency of

the sequence up to token $t-1$. Formally, let:

- $\log P_{\text{model}}(y^i_{1:t})$: cumulative log-probability from the model,
- $\log P_{\text{clf}}(y^i_{1:t-1})$: cumulative classifier log-probability indicating factual consistency of the sequence,
- $\lambda \in [0, 1]$: weighting factor determining classifier influence.

The adjusted beam score for each hypothesis is then:

$$S(y^i_{1:t}) = (1-\lambda) \cdot \log P_{\text{model}}(y^i_{1:t}) + \lambda \cdot \log P_{\text{clf}}(y^i_{1:t-1})$$

This formula ensures that the decoding process not only favors sequences that are likely according to the language model, but also those that are evaluated as truthful by the classifier. Notably, the classifier operates on the previous partial sequence $y^i_{1:t-1}$ rather than including the current token. This design choice avoids evaluating all *beam width* $\times$ *beam width* possible continuations, which would grow exponentially, preserving the computational efficiency of the standard beam search. By integrating factuality into the decoding process, Beam+Guided Decoding strikes a balance between diversity, fluency, and factual accuracy, making it well suited for applications prone to hallucination, such as summarization and question answering.

4. Experiments

4.1 Research problems

This chapter presents a set of experiments designed to evaluate the effectiveness of classifier-guided decoding methods. The primary objective is to assess how these strategies perform in mitigating hallucinations in large language models (LLMs), with a focus on contextual tasks such as question answering (QA) and summarization (SUMM). These tasks are selected because of their high factual requirements and the frequent occurrence of content that is seemingly correct but, in fact, incorrect in model-generated outputs. To provide a comprehensive comparison, the proposed classifier-guided decoding methods are evaluated alongside widely used baseline strategies, including greedy decoding, beam search, and DoLa. This comparison aims to highlight trade-offs in factual consistency and computational diversity between standard decoding techniques and hallucination-aware approaches.

The experiments are structured around the following research questions:

RQ1: How effectively do hallucination-aware decoding strategies reduce contextual hallucinations compared to standard decoding methods?

RQ2: How well do hallucination-aware decoding strategies generalize across different tasks?

RQ3: What are the computational trade-offs (in GPU memory and decoding latency) of reducing hallucinations using classifier-guided methods?

RQ4: How sensitive is hallucination reduction to the configuration of classifier-guided parameters?

RQ5: How does the nature of the underlying dataset influence the effectiveness of hallucination reduction strategies?

All experiments were conducted using the Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH), under computational grant no. PLG/2024/017840. I gratefully acknowledge their resources and technical support, which made the large-scale training and evaluation feasible.

4.2 Experimental Setup

To evaluate classifier-guided decoding methods, it was necessary to first train hallucination detection classifiers according to the methodology proposed in [20]. These classifiers were later adapted to support the novel decoding strategies proposed in this work.

4.2.1 Data and Models

To evaluate the generalization capability of decoding strategies, two primary contextual tasks were selected:

- **Question Answering (QA):** Natural Questions (NQ) [17] and HotPotQA [37].
- **Summarization (SUMM):** CNN/Daily Mail (CNN/DM) [28] and XSum [22].

Each dataset was randomly split, with 100 examples held out for decoding evaluation. The CNN/DM and NQ datasets were utilized to train hallucination classifiers, features constructed by aggregation technique of attention scores named AggTruth SUM [20]. To evaluate **robustness and generalization across domains** and tasks, classifiers trained on QA (e.g., NQ) were tested on a different QA dataset (e.g., HotPotQA) and across-task on summarization datasets (e.g., CNN/DM, XSum), and vice versa. This setup enables the assessment of both domain transfer and task transfer capabilities. To evaluate model-agnostic applicability of the proposed decoding strategies, experiments were conducted using a diverse set of publicly available small- and medium-scale language models:

- `meta-llama/Llama-2-7b-chat-hf` [32]
- `unsloth/gemma-2-9b-it-bnb-4bit` [30]
- `microsoft/Phi-3.5-mini-instruct` [1]

This selection enables the evaluation of decoding methods across varying model architectures, parameter scales, and training paradigms.

4.2.2 Pipeline for Classifier Training

To facilitate the development of hallucination detection models, a structured pipeline was implemented for classifier construction, as outlined in Figure 4.1 from [20]. The pipeline begins by generating model responses and extracting the corresponding attention maps. These responses are then submitted to GPT-4o [23] for annotation of hallucinations at the token level. Each token is aligned with a binary label: 1 if hallucinated, 0 otherwise. The utilization of GPT-4o for annotation is supported by recent studies [6, 20], which highlight its high level of agreement with human evaluators. The annotation step is crucial, as it forms the basis of the supervised learning task. To ensure the quality of

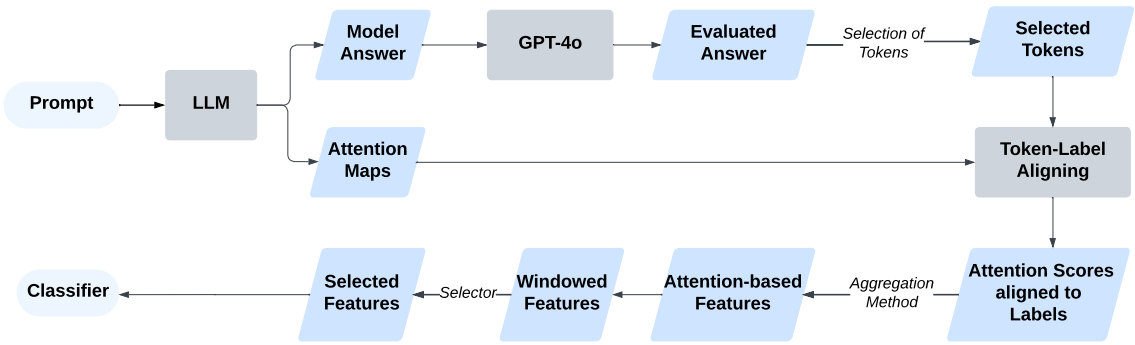


Figure 4.1: The full end-to-end pipeline [20]: from generating responses and extracting attention to selecting features and training the hallucination detection classifier.

these annotations, a subset was cross-validated against human judgments, achieving a Cohen's Kappa of 0.7 [20], indicating substantial agreement. This validation reinforces the viability of automated annotation pipelines in hallucination research.

Following annotation, attention scores corresponding to passage tokens are aggregated using the AggTruth SUM methodology [20]. This method computes the total attention score of the tokens generated on the context, aligning with the hypothesis that a higher context focus correlates with factual responses. The generated sequences are divided into token overlapping windows (in this study window is equal to 1), following the design suggested in [6, 20]. A window is labeled as hallucinated if it contains at least one hallucinated token. For each window, attention features are averaged across tokens, producing a single feature vector per attention head. All examples are then concatenated into a unified dataset. A selector module is used to retain the most informative features and the resulting data is used to train a hallucination classifier. This classifier is then calibrated and integrated into the language model's decoding process to reduce the generation of hallucinated content during inference. Compared to approaches such as Lookback Lens [6], which analyzes broader prompt-level attention, the AggTruth method focuses specifically on context attention and provides finer granularity. The pipeline supports plug-and-play integration with any transformer model that provides attention maps, making it applicable across diverse architectures.

4.2.3 Pipeline for Decoding Strategies Evaluation

To evaluate the effectiveness of different decoding strategies in mitigating hallucinations, a dedicated benchmarking pipeline was implemented. The evaluation process consisted of two primary components: (1) factual consistency assessment using an external LLM as a judge, and (2) computational performance profiling, including GPU memory usage and generation speed.

First, responses were generated using various decoding strategies, including both standard methods (e.g., greedy decoding, beam search, DoLa) and classifier-guided approaches. For each strategy, answers were generated on a fixed sample of QA and summarization prompts. Then these responses were submitted to GPT-4o, which acted

as an external evaluator, labeling each response hallucinated or not. This enabled the computation of accuracy scores that represent the percentage of hallucination-free outputs per strategy and dataset.

In parallel, each decoding strategy was evaluated in terms of computational efficiency. For each prompt, GPU memory consumption and total generation time were recorded. Time per generated token was calculated to assess decoding latency. This allowed for a direct comparison of computational trade-offs across strategies.

To ensure consistency, all models were executed with controlled settings, using uniform prompt templates and hyperparameters. Furthermore, GPU memory statistics were reset and collected prior to each decoding run. This benchmarking framework provides a comprehensive view of both the factual quality and efficiency of each decoding method, underscoring their viability in real-world deployment scenarios.

4.3 Results and Analysis

Following the evaluation methodology, I evaluated three models using a range of decoding strategies: Guided decoding, Lookback-Guided decoding, Beam Search, Beam+Guided, Self-Reflection, and Finetuned variants. The results are analyzed in relation to the research questions defined earlier. Each classifier used in decoding step is only trained on one dataset to test same task and cross-performance of strategies.

It is important to note that Guided Decoding and Self-Reflection rely on manipulating the model's internal key-value cache, which is not feasible in architectures with sliding attention, such as Gemma2. For this reason, these strategies are excluded from the corresponding evaluation.

Before diving into the individual research questions, I would like to clarify an important implementation detail. The standard Guided decoding was applied using a sliding window of size 1 token, which made it lightweight and compatible with all decoding strategies that work on **token level**. In contrast, Lookback-Guided decoding used a broader window of 8 tokens, aggregating attention features across a larger context. Additionally, incorporating Lookback into models other than Llama proved to be challenging, which limited its evaluation.

RQ1: Effectiveness of hallucination-aware decoding

To evaluate how effectively hallucination-aware decoding strategies mitigate contextual hallucinations, I compared the best and average accuracy scores across different decoding methods and model backbones (Gemma2, Llama2, and Phi-3.5). Accuracy here is defined as the proportion of hallucination-free outputs, as judged by GPT-4o.

For Gemma2 (Figure 4.2), DoLa achieved the highest accuracy (0.825), followed by Beam+Guided and Greedy decoding, both reaching 0.810. In particular, Gemma2 was the only model equipped with a sliding attention mechanism, restricting the use of other guided decoding strategies. Overall, most of the methods performed consistently well in this architecture and the choice of decoding strategy had a minimal impact on accuracy. Interestingly, the Fine-Tuned variant reached only 0.725 accuracy, supporting the observation that finetuned models may overfit and fail to generalize across different tasks than those trained on.

For Llama2 (Figure 4.3), Beam+Guided and Fine-Tuned variants both achieved the highest accuracy (0.643). Although classifier-guided methods demonstrated lower overall performance, they consistently outperformed Greedy decoding across all settings. In particular, Guided Decoding, which operates **at the token level** and was trained using AggTruth, **outperformed** the Lookback Guided method, which relies on an 8 token window. This suggests that operating at the token level can be effective in reducing hallucinations. These results suggest that **hallucination-aware strategies are particularly beneficial for models like Llama2 that may not possess strong factual alignment**.

Interestingly, for Phi-3.5 (Figure 4.4), Beam+Guided achieved the highest accuracy (0.707), closely followed by Beam Search (0.705) and SelfRef (0.697). Surprisingly, Greedy decoding also performed competitively (0.690), nearly matching Beam Search

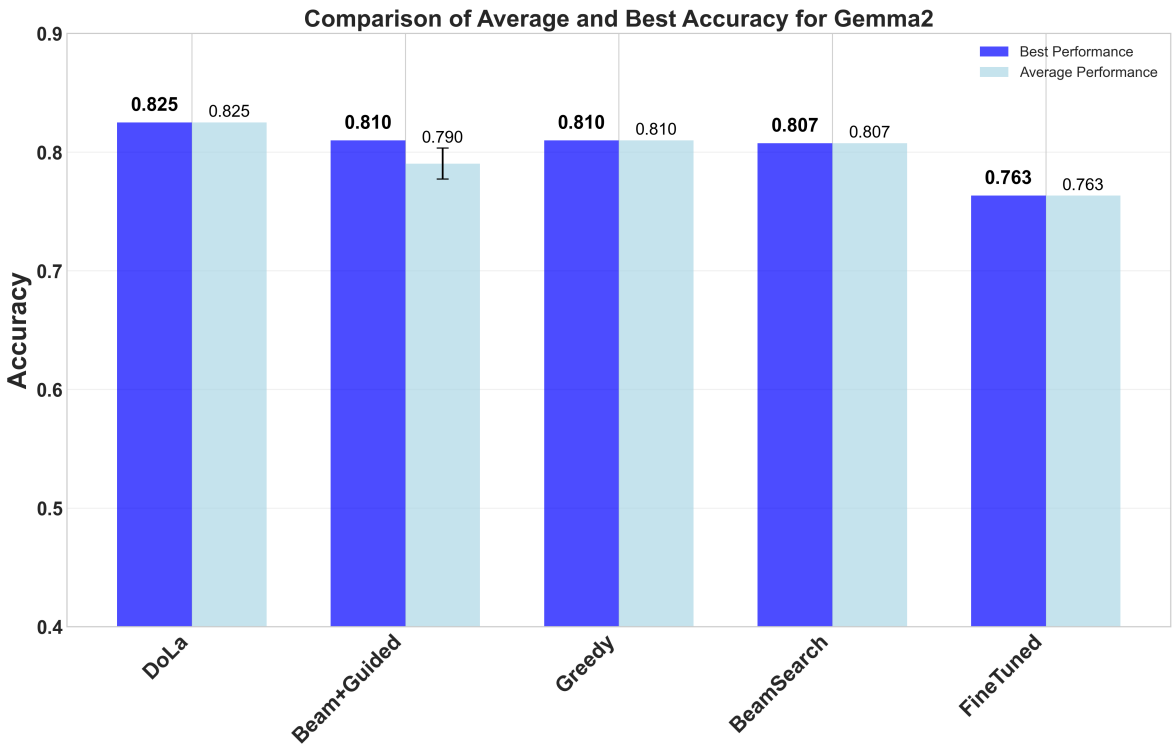


Figure 4.2: Comparison of best and average accuracy across decoding methods for Gemma2. Classifier-guided approaches like Beam+Guided and DoLa consistently outperform standard decoding baselines.

and surpassing several hallucination-aware strategies. These results suggest that the effectiveness of classifier-guided decoding is highly dependent on the underlying model architecture. Methods such as GuidedDec, DoLa, and even the Fine-Tuned variant exhibited lower performance, indicating a potential lack of transferability of some hallucination mitigation techniques to models like Phi-3.5.

To better account for the intrinsic differences in baseline capabilities between models, Figure 4.5 illustrates the relative improvement in factual accuracy for each decoding strategy over the Greedy baseline acro]ss three models. This comparison helps reveal how effectively each method enhances factuality without being confounded by the models' inherent differences in raw performance.

As shown in Figure 4.5, Beam+Guided demonstrates the highest and most consistent improvements in factual accuracy over the Greedy baseline, most notably for Llama2, where it achieves a gain exceeding 25%. This substantial margin highlights the effectiveness of combining structured search with external guidance in models that are prone to hallucinations. BeamSearch also shows solid gains, particularly in Llama2, with moderate improvements in Phi-3.5, strengthening its role as a strong baseline for hallucination-aware decoding. In contrast, decoding strategies such as DoLa, Guided Decoding, and FineTuned often fail to outperform Greedy, especially in Phi-3.5. DoLa, in particular, suffers a substantial performance drop (over 10%) in this model, which significantly lowers its overall utility. FineTuned decoding underperforms Greedy in both

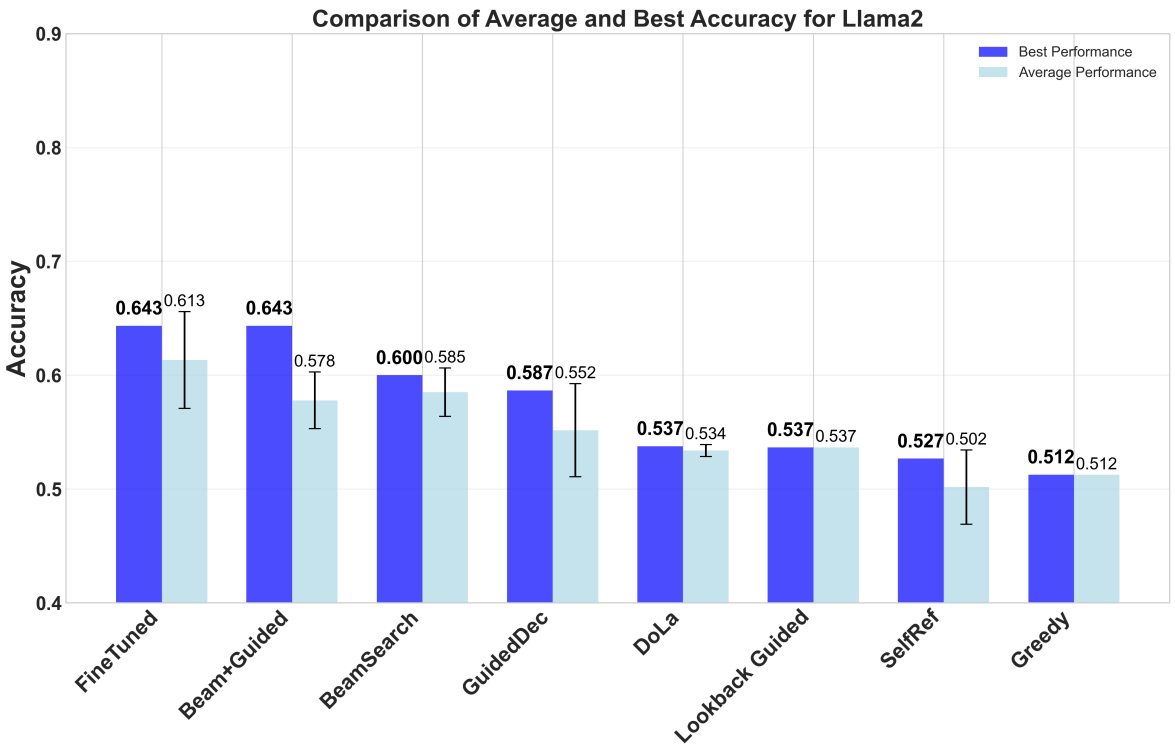


Figure 4.3: Comparison of best and average accuracy across decoding methods for LLaMA2. Beam+Guided and Fine-Tuned decoding outperform traditional approaches like Greedy.

Gemma2 and Phi-3.5, suggesting overfitting to a specific domain. GuidedDec achieves modest improvements, but they are not consistent across all models, while SelfRef offers slight but positive gains, with results varying based on the underlying model's architecture. It is worth noting that Beam+Guided, as an extension of Beam Search, and SelfRef, building upon Greedy decoding, both **outperformed their respective base strategies**. This suggests that enhancing standard decoding methods with external guidance mechanisms can lead to significant improvements in factual accuracy.

Overall, these findings suggest that hybrid decoding approaches that combine structured search with hallucination-aware guidance offer robust **improvements in hallucination mitigation**, particularly **when they are well-matched to the underlying model architecture**.

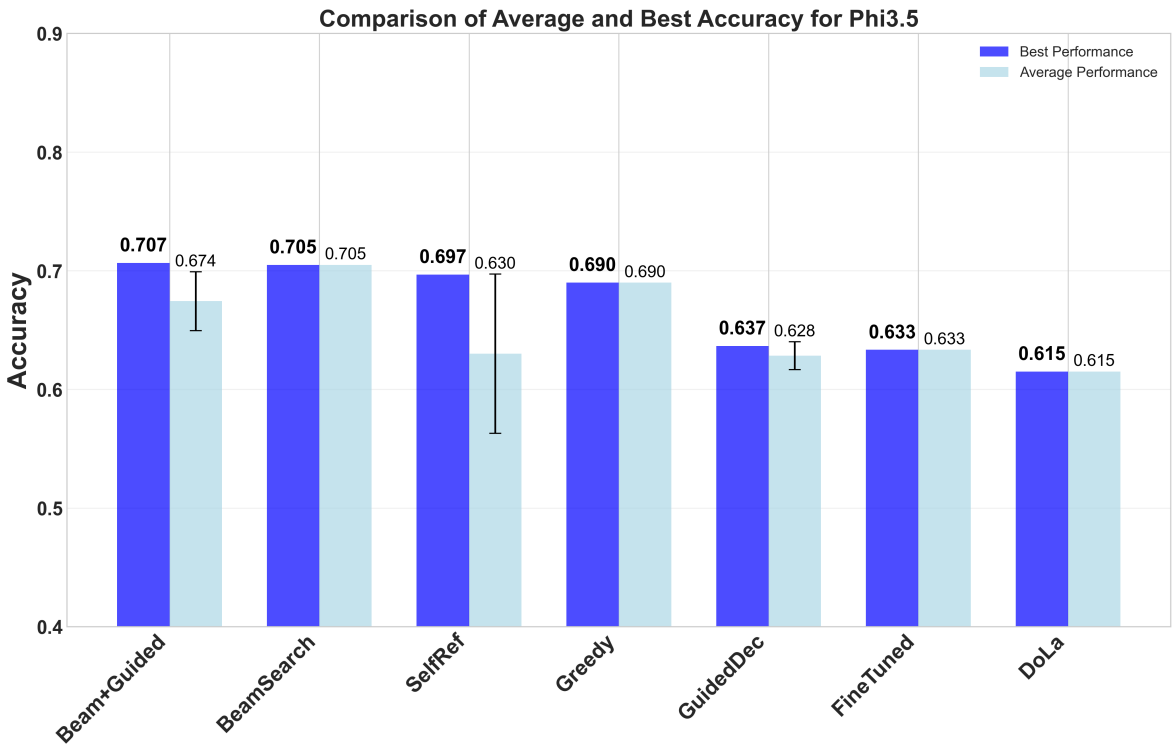


Figure 4.4: Comparison of best and average accuracy across decoding methods for Phi-3.5.

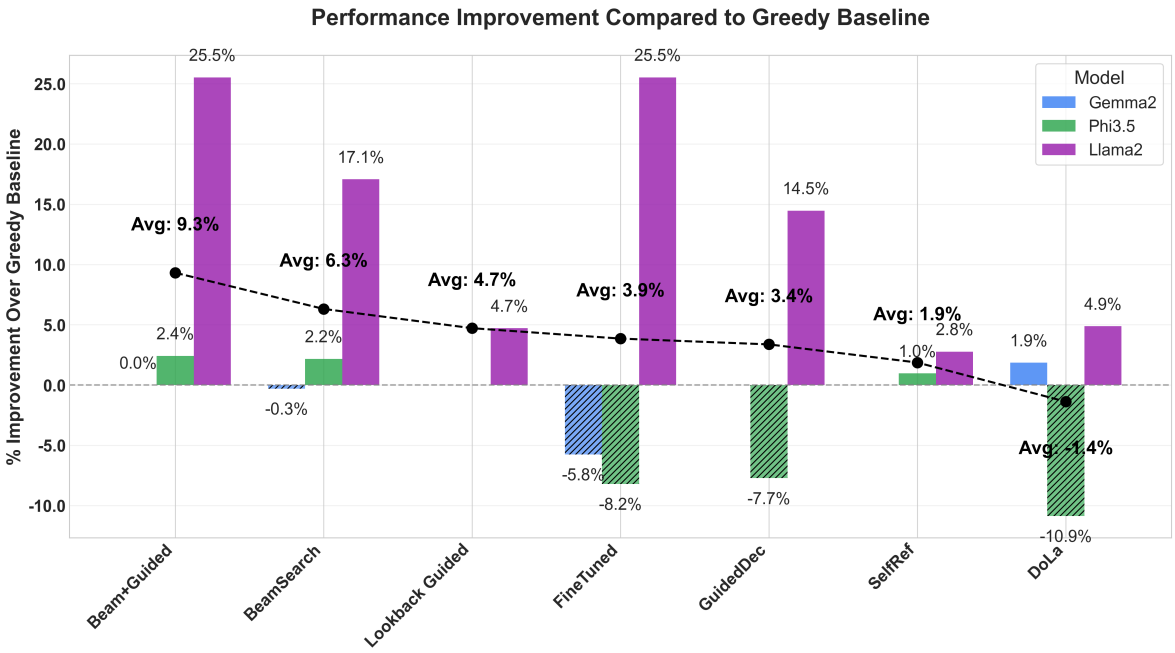


Figure 4.5: Relative improvement in factual accuracy over each model's Greedy baseline for Gemma2, Phi-3.5, and Llama2. Average improvements across models are annotated with a dotted trend line to highlight cross-model generalizability.

RQ2: Generalization of decoding strategies across tasks

To assess how well hallucination-aware decoding strategies generalize across different types of contextual tasks, I compared their performance on two distinct benchmarks: Question Answering (QA) and Summarization. Although both tasks require factual correctness, they differ in structure: QA typically expects concise and precise answers, while summarization involves compressing longer inputs into coherent and factually accurate summaries.

As shown in Figure 4.6, most decoding strategies generally generalize reasonably well across both QA and summarization, although some task-specific patterns are evident. In particular, Llama2 shows a clear difference in performance between tasks: It performs significantly better at summarization than at QA, regardless of the decoding strategy. This may be due to the weaker factual alignment capabilities of Llama2 being more exposed in the precision-demanding QA setting. In contrast, Gemma2 shows stronger performance on QA. This suggests that Gemma2 is better equipped for direct factual retrieval tasks. For Phi-3.5, the performance remains more balanced between tasks, with no major divergence, indicating a more uniform behavior between summarization and QA.

Despite these task-level differences, strategies like Beam+Guided and Beam Search performed consistently well across both tasks, showing only minimal drop-offs. This highlights their robustness and adaptability to varying factual demands.

Interestingly, the Fine-Tuned variant tended to perform best on the task it was specifically trained on, reinforcing that finetuning offers task specialization, but possibly at the expense of broader generalization.

Taken together, these findings indicate that while hallucination-aware decoding strategies generalize reasonably across tasks, their relative effectiveness is influenced by the nature of the task and the underlying alignment properties of the model. Hybrid methods that balance exploration with factual filtering, such as Beam+Guided, appear stable across diverse settings.

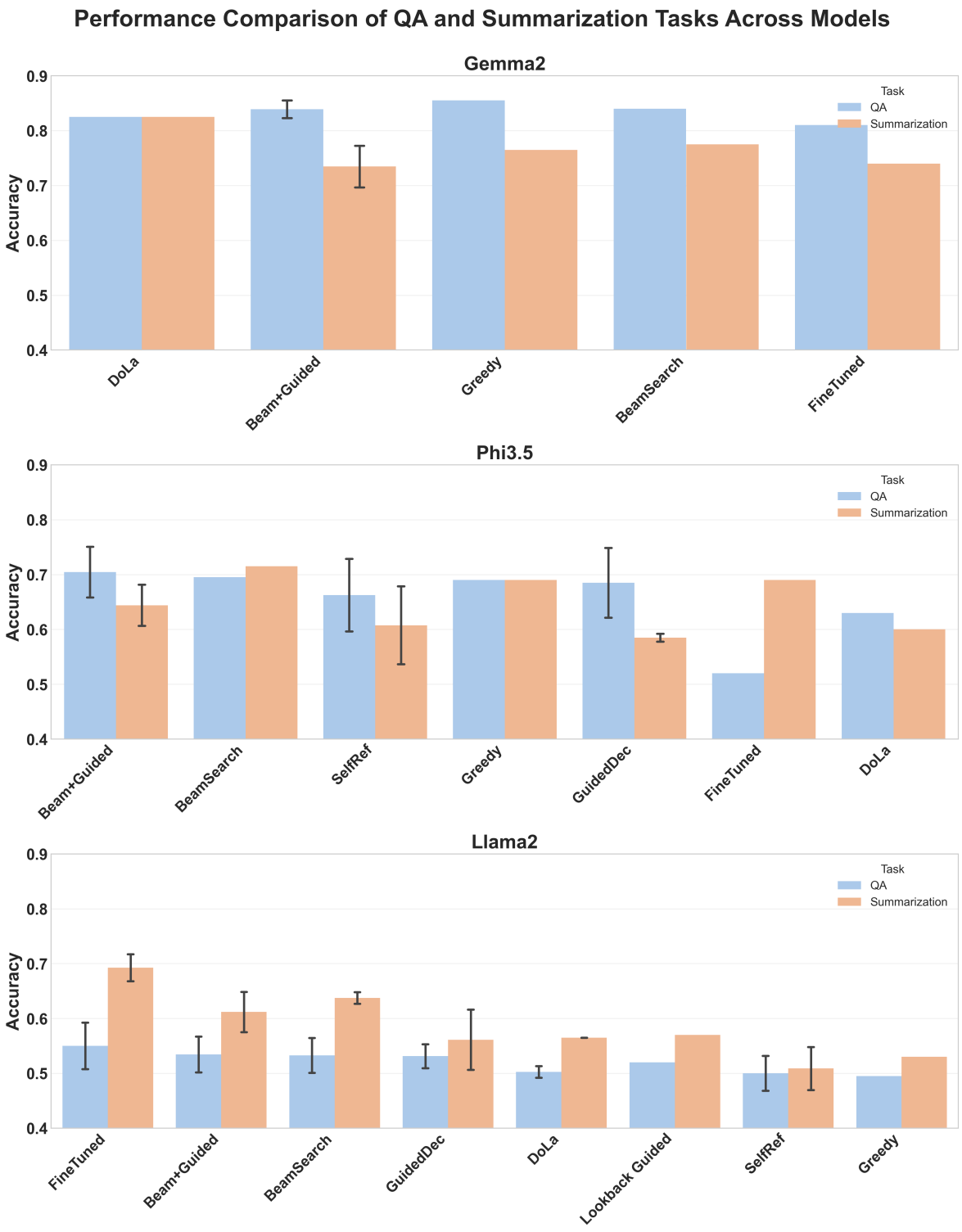


Figure 4.6: Performance of decoding strategies on QA vs. Summarization for (Llama2, Gemma2 and Phi3.5). Accuracy measured as hallucination-free response rate.

RQ3: Computational trade-offs of classifier-guided methods

To evaluate the practical feasibility of hallucination-aware decoding, I analyzed the computational demands of each strategy in terms of GPU memory usage and decoding latency. These metrics are especially relevant when deploying LLMs at scale or in real-time applications.

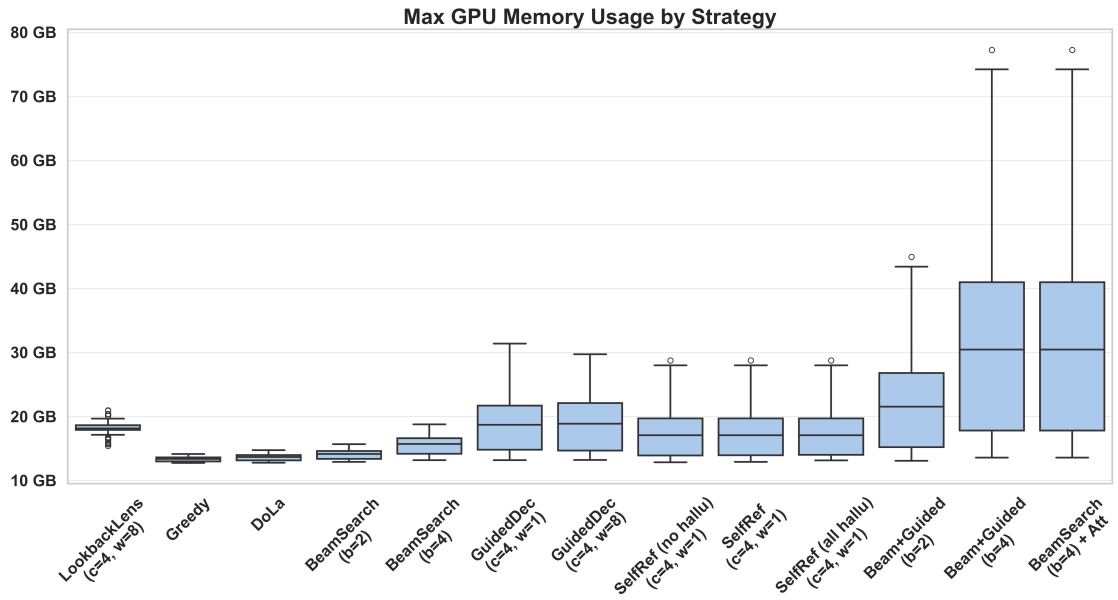


Figure 4.7: Maximum GPU memory usage by decoding strategy. Classifier-guided methods—especially when combined with beam search—show increased memory requirements.

As shown in Figure 4.7, standard methods such as Greedy and DoLa demonstrated the lowest memory usage, typically under 15 GB. In contrast, classifier-guided decoding—particularly Beam+Guided with a beam size of 4—reached significantly higher memory footprints, occasionally exceeding 70 GB. This elevated memory demand is mainly due to the need to return and store attention weights from the forward pass of the model, which are required for hallucination scoring. Unlike standard inference where attention maps would be discarded after use, in this case they must be explicitly retained in GPU memory, making the process more memory intensive.

It is important to note that this memory overhead is not intrinsic to classifier-guided decoding itself, but rather to the attention matrices needed by the classifier. For comparison, standard BeamSearch with attention return enabled exhibited nearly identical memory usage. Consequently, the increase in memory consumption is a result of the instrumentation requirement, rather than a fundamental inefficiency of the method.

Figure 4.8 presents the decoding latency, expressed as time per generated token. Here, Greedy decoding and DoLa again emerged as the most efficient strategies, averaging below 60 ms/token. More complex approaches such as GuidedDec and Self-Reflection showed moderate increases, while Beam+Guided decoding with larger beam widths significantly increased generation latency, in some cases exceeding 300 ms/token. Notably,

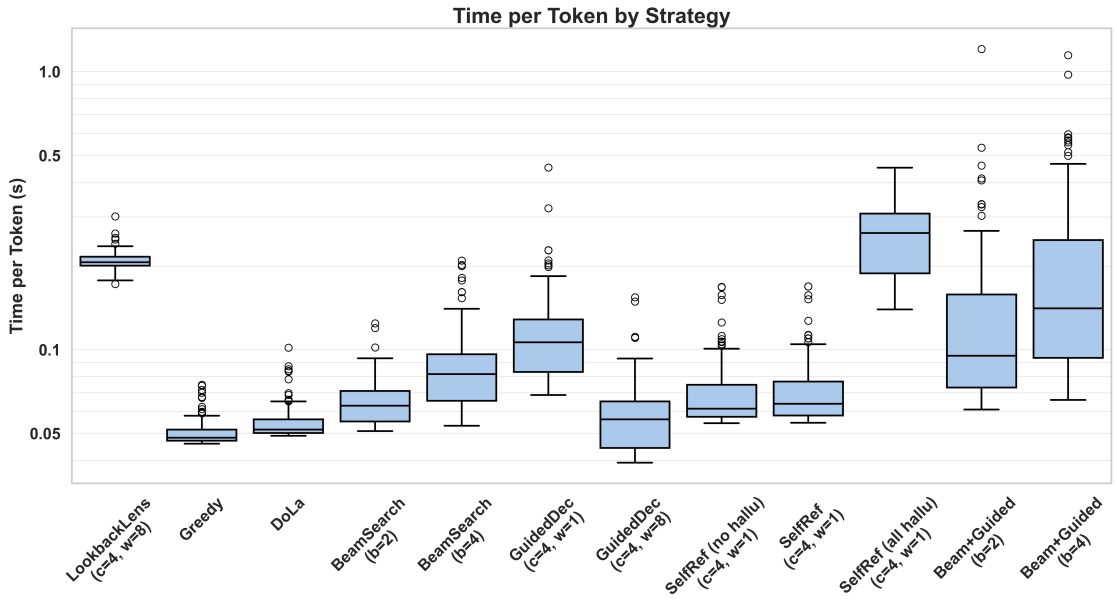


Figure 4.8: Decoding latency (time per token) by strategy. Lightweight methods such as Greedy and DoLa remain the fastest. Classifier-guided and beam-based methods show increased generation times.

Self-Reflection decoding showed a correlation between the number of hallucinations detected and increased generation time, due to repeated evaluations and scoring steps. GuidedDec, a modification of Lookback Lens Decoding, achieves faster and more efficient generation, showing a latency time that is three times shorter.

In summary, while classifier-guided decoding effectively reduces hallucinations, it introduces **significant computational overhead**. The memory and latency costs are mainly due to the need to extract and retain attention-level features for classification, rather than inefficiencies in the decoding itself. This makes such methods more suitable for **offline generation, batch processing, or high-resource environments**. For applications constrained by latency or resource limitation, simpler methods, such as DoLa, provide a favorable balance between efficiency and factuality.

RQ4: Sensitivity to classifier-guided decoding parameters

Since both Self-Reflection decoding and Beam+Guided strategies introduce tunable parameters, a sensitivity analysis was performed to assess how these parameters affect overall hallucination reduction performance. Specifically, I analyzed the effect of the hallucination threshold in SelfReflection and the α value (which balances classifier scores with beam probabilities) in Beam+Guided decoding.

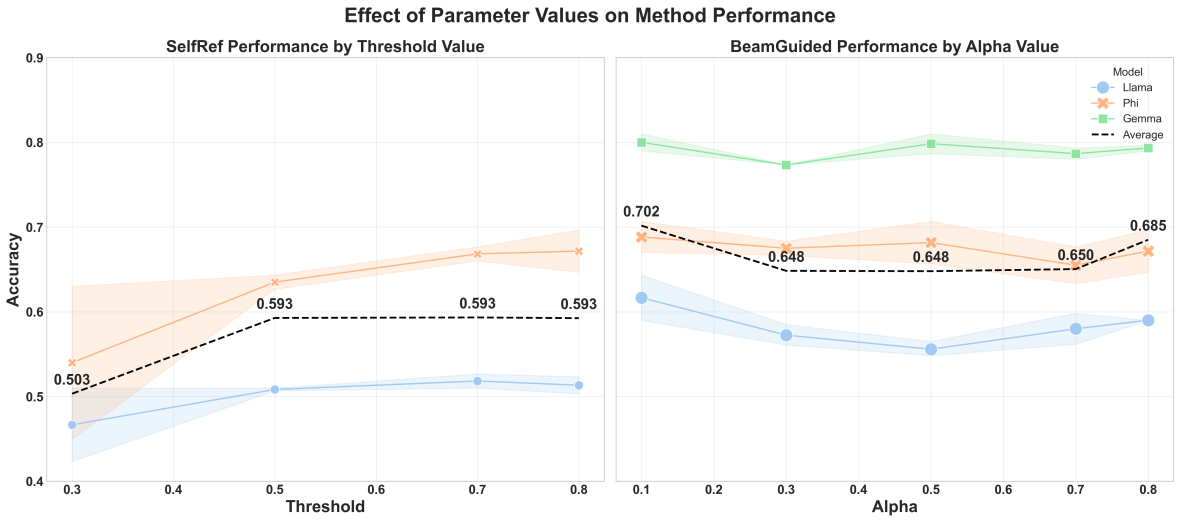


Figure 4.9: Effect of classifier-guided decoding parameters on accuracy. Left: SelfRef performance across classification thresholds. Right: Beam+Guided performance across different α values.

Figure 4.9 shows that classifier-guided methods exhibit moderate but significant sensitivity to threshold settings. For SelfReflection (left panel), increasing the threshold from 0.3 to 0.7 improves accuracy, particularly for the Phi model, with performance plateauing around 0.7–0.8. Interestingly, a threshold of 0.5 already provides near-optimal results, suggesting that calibrated models produce reliable confidence scores that align well with an intuitive decision boundary.

For Beam+Guided decoding (right panel), the α parameter controls the weight given to hallucination scores during beam selection. Interestingly, performance was highest at the extremes, both low ($\alpha = 0.1$) and high ($\alpha = 0.8$) values, while intermediate values (for example, $\alpha = 0.3$ to 0.5) showed a dip in accuracy. This pattern was consistent across models, indicating that effective decoding tends to rely either primarily on the beam log-probabilities or on the classifier log-scores, but not on a balanced combination of both. The exception was the Gemma model, where the best results were achieved with a near-equal weighting (50/50) of the two scores.

RQ5: Influence of the underlying (training) dataset on effectiveness

To assess how the characteristics of the training dataset impact the effectiveness of classifier-guided hallucination-aware decoding, I compared average factual accuracy across two distinct task domains: summarization (CNN/DM) and question answering (Natural Questions). These domains differ in both format and the type and density of factual content they present.

Figure 4.10: Average factual accuracy across CNN/DM (summarization) and NQ (question answering) datasets.

As shown in Figure 4.10, the choice of training dataset does influence the decoding performance, although the effect is generally modest and varies by model architecture. For example, Gemma2 shows stable accuracy in both datasets, while Phi3.5 and Llama2 perform slightly better when trained on the QA-based dataset (NQ). In particular, **training on a data set aligned with the weakest task of a model appears to yield greater improvements**, suggesting that dataset selection can be strategically tailored to compensate for underrepresented capabilities in the base model.

These results suggest that the classifier relies on generalizable domain and invariant features, aggregated attention patterns, to identify hallucinations. Consequently, its effectiveness transfers reasonably well across tasks with different content structures, reducing reliance on the specifics of the training domain.

To synthesize the above findings, Figure 4.11 presents a scatter plot summarizing the trade-offs between accuracy, GPU memory usage, and decoding latency for various strategies on the Llama2 model. The x-axis represents decoding latency (time per token), the y-axis shows accuracy, and the bubble size corresponds to GPU memory consumption. The visualization clearly illustrates the strengths and weaknesses of each method. Beam+Guided decoding stands out with the highest accuracy but incurs the steepest computational cost, particularly in memory and latency. In contrast, lightweight methods such as Greedy and DoLa maintain low resource usage but deliver comparatively lower factual performance. Strategies like GuidedDec and BeamSearch strike a middle ground, offering meaningful hallucination reduction while keeping overhead relatively manageable. This holistic comparison reinforces that no single method dominates across all criteria, and the choice of decoding strategy should be guided by application-specific trade-offs between factual reliability and computational efficiency.

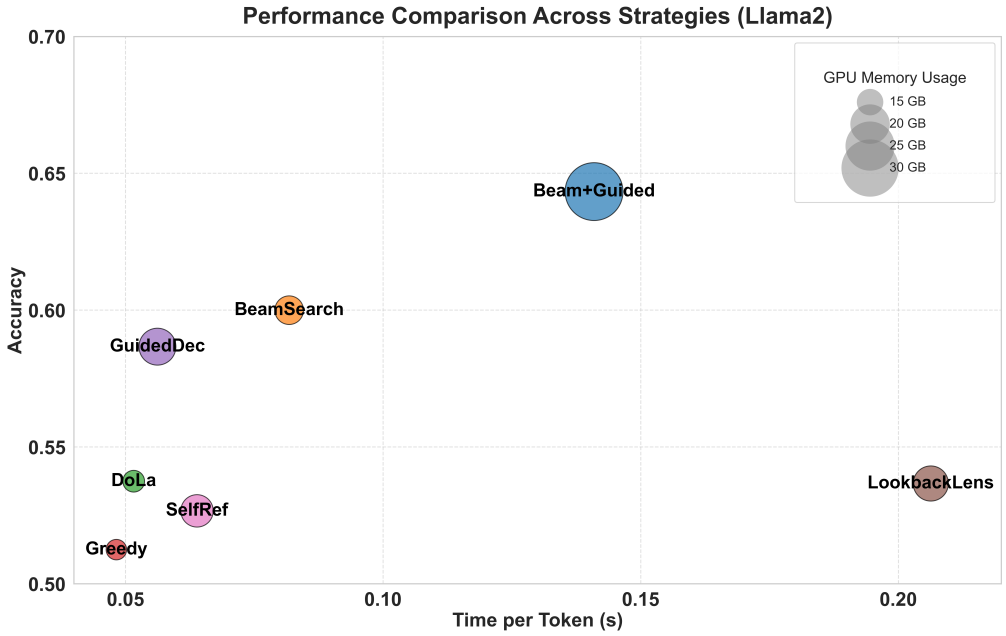


Figure 4.11: Performance comparison across decoding strategies on Llama2. Accuracy is plotted against GPU memory usage, with bubble size reflecting decoding latency (time per token).

5. Conclusions

This thesis investigated the potential of classifier-guided decoding strategies as a method of reducing hallucinations in large language models (LLMs), particularly in contextual tasks like question answering and summarization. Hallucinations, outputs that diverge from the provided context despite accurate input, remain a **critical challenge in real-world deployments**. As argued by Banerjee et al., hallucinations in LLMs are an inherent limitation of current generative paradigms, and while they cannot be fully eliminated, we can develop mechanisms to manage and mitigate their impact [3]. The proposed methods integrate attention-based hallucination classifiers into the decoding process, improving factual reliability without modifying the model architecture or requiring fine-tuning. Three novel strategies were introduced: **Improved Guided Decoding, Self-Reflection Decoding, and Beam+Guided Decoding**. Each of these approaches incorporates classifier scores during the decoding process, enabling the model to prioritize more contextually grounded alternatives. Improved Guided Decoding builds on Lookback methods using caching and batched inference to reduce overhead. Self-Reflection Decoding revises flagged tokens using internal attention signals. Beam+Guided Decoding combines beam search with classifier scoring for more truthful responses.

Experiments across models (Llama2, Gemma2, Phi-3.5) and tasks showed consistent improvements over baseline decoding, with Beam+Guided decoding proving to be the most robust. Although some strategies were incompatible with sliding attention, overall factual gains were consistent and task-agnostic. These benefits come with trade-offs. Beam+Guided decoding increases memory use and latency due to the need for attention feature extraction, not due to the decoding logic itself. Figure 4.11 visualizes these trade-offs, showing how the proposed methods compare in terms of decoding speed, memory footprint, and factual accuracy for the Llama2 model. While Beam+Guided decoding exhibits higher memory and latency than simpler methods like Greedy, it delivers superior accuracy, making it a compelling choice for scenarios where factual quality is paramount. The radar chart illustrates the central claim of this thesis: **that the reduction of contextual hallucination through classifier-guided decoding is an effective way to improve the reliability of the response in large language models (LLMs).**

In conclusion, this work introduces modular, interpretable, and model-agnostic decoding strategies that reduce contextual hallucinations at inference time, ideal for deployment scenarios where retraining is infeasible. Despite resource demands, these methods are well suited for **offline generation, batch processing, and high-resource or critical environments such as hospitals or legal institutions**, where improved factuality outweighs added cost.

Future Work

The most promising direction is improving attention retrieval under memory constraints. Enhancing retrieval mechanisms could increase the applicability of the proposed methods by reducing their resource requirements. Techniques such as Self-Reflection Decoding currently depend on cache manipulation, which poses challenges when used alongside sliding attention mechanisms. Refactoring these methods to integrate more seamlessly with memory-efficient caches is a worthwhile route for exploration. Another area for exploration is the use of classifiers based on hidden states rather than attention maps, which may offer more compatibility with memory-efficient inference. Applying the proposed decoding strategies to smaller models, particularly those with fewer than one billion parameters, is also worth investigating. Such studies could shed light on their effectiveness in edge scenarios or real-time systems, where computational efficiency is essential. The development of unsupervised or self-supervised alternatives to hallucination classifiers presents another valuable opportunity. These methods could reduce reliance on labeled data by leveraging intrinsic uncertainty estimates, entropy-based heuristics, or contrastive learning techniques to guide generation. Finally, future work may explore architectural modifications to the models themselves. Integrating inductive biases or structures that enhance context fidelity, such as attention mechanisms that dynamically penalize divergence from the input, could improve reliability beyond inference-time patching. Collectively, these directions lay the foundation for more context-aware and trustworthy language models.

5. Bibliography

[1] M. Abdin, J. Aneja, H. Awadalla, S. Bubeck, M. Cai, A. D. G. Chase, M. Dixon, R. Eldan, E. Haider, J. Hao, S. A. Jacobs, A. Mitra, H. Modi, A. Nguyen, R. Pryzant, and R. Ward. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, abs/2404.14219, 2024. First 15 authors listed; original posting April 22 2024.

[2] A. Azaria and T. Mitchell. The internal state of an llm knows when it's lying. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore, Dec. 2023. Association for Computational Linguistics.

[3] S. Banerjee, A. Agarwal, and S. Singla. Llms will always hallucinate, and we need to live with this. *arXiv preprint arXiv:2409.05746*, abs/2409.05746, Sept. 2024. Published September 9, 2024; introduces "Structural Hallucination," argues hallucinations are mathematically inevitable :contentReference[oaicite:0]index=0.

[4] P. Bojanowski, É. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, and R. Child. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 1877–1901, Vancouver, Canada, Dec. 2020. Curran Associates, Inc.

[6] Y. Chuang, L. Qiu, C. Hsieh, R. Krishna, Y. Kim, and J. R. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In Y. Al-Onaizan, M. Bansal, and Y. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1419–1436, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics.

[7] Y. Chuang, Y. Xie, H. Luo, Y. Kim, J. R. Glass, and P. He. Dola: Decoding by contrasting layers improves factuality in large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, Online, Jan. 2024. ICLR. Oral presentation.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, Online, Apr. 2021. Oral presentation.

[9] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, June 2024.

[10] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig. Does fine-tuning llms on new knowledge encourage hallucinations?, 2024.

[11] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. Technical Report Google DeepMind Blog, DeepMind, Nov. 2014. Originally released as a technical blog report; widely cited via the arXiv preprint.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[14] Z. Ke, H. Lin, Y. Shao, H. Xu, L. Shu, and B. Liu. Continual training of language models for few-shot learning. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, pages 10205–10216, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[15] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz, A. Kocoń, B. Koptyra, W. Mieleszczenko-Kowszewicz, and P. Miłkowski. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, 2023.

[16] T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In E. Blanco and W. Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[17] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[18] H. Li, L. Ding, M. Fang, and D. Tao. Revisiting catastrophic forgetting in large language model tuning. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Findings*

of the Association for Computational Linguistics: EMNLP 2024, pages 4297–4308, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics.

[19] P. Manakul, A. Liusie, and M. J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, pages 9004–9017, Singapore, Dec. 2023. Association for Computational Linguistics.

[20] P. Matys, J. Eliasz, K. Kiełczyński, M. Langner, T. Ferdinan, J. Kocoń, and P. Kazienko. Aggtruth: Contextual hallucination detection using aggregated attention scores in llms. In *International Conference on Computational Science*. Springer, 2025. Accepted for publication.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations Workshop Track*, Scottsdale, AZ, Feb. 2013. Originally appeared as arXiv:1301.3781.

[22] S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807, Brussels, Belgium, Oct.–Nov. 2018. Association for Computational Linguistics.

[23] OpenAI. GPT-4o: Openai's new multimodal model. https://openai.com/index/gpt-4o, May 2024. Accessed: 2025-05-08.

[24] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, pages 8748–8763, Virtual Conference, July 2021. PMLR.

[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical Report (OpenAI), OpenAI, Apr. 2019. Accessed: 2025-05-09.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[28] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. In R. Barzilay and M. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[29] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[30] G. Team, M. Rivière, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, J. Ferret, P. Liu, P. Tafti, A. Friesen, and M. Casbon. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024.

[31] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016)*, pages 2469–2474, Cancún, Mexico, Dec. 2016. IEEE. Also appeared as arXiv 1709.01686.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, É. Grave, and G. Lample. Llama: Open and efficient foundation language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*, Online, Apr. 2023. OpenReview.net. Oral presentation.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 5998–6008, Long Beach, CA, USA, Dec. 2017. Curran Associates, Inc.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, BC, Canada, Feb. 2018. OpenReview.net. Accepted as poster.

[35] S. Woo, J. Park, J. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Proceedings of the European Conference on Computer Vision (ECCV 2018)*, volume 11211 of *Lecture Notes in Computer Science*, pages 3–19, Munich, Germany, Sept. 2018. Springer International Publishing.

[36] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, and

Ł. Kaiser. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint*, abs/1609.08144, 2016. Submitted Sep 26 2016; widely known as GNMT.

[37] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380, Brussels, Belgium, Oct.–Nov. 2018. Association for Computational Linguistics.

[38] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, and J. Jiang. A survey of large language models, 2023.

List of Figures