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

Field of study: **Artificial Intelligence (SZT)**

MASTER THESIS

Comparison of Hidden State-Based and Attention Map-Based Methods for Detecting Contextual Hallucinations in LLMs

Jan Eliasz

Supervisor
PhD Jan Kocoń

Keywords: large language models, hallucinations, hidden states, attention maps

WROCŁAW 2025

Abstract

The phenomenon of hallucinations in large language models (LLMs) remains a significant barrier to their deployment across a wide range of real-world applications, where a potential error may lead to serious consequences, such as in the medical or legal domain. Numerous methods have been proposed to address the task of detecting hallucinations in model responses. This work compares two major approaches to hallucination detection that leverage the internal state of an LLM: methods based on hidden states and methods based on attention maps. The comparison is conducted across a diverse set of experimental scenarios, including different tasks, datasets, response chunking strategies, and hyperparameter settings. The results show that neither approach consistently outperforms the other in all settings, highlighting the need for hallucination detection methods that are in accordance with the specific characteristics of the task. Hidden state-based methods detect hallucinations better when evaluating full responses and in question answering scenarios. In contrast, attention map-based methods are more effective in summarization tasks and when processing chunked responses. That said, the method demonstrating the strongest generalization capabilities is *AggTruth* (attention map-based one), which is therefore recommended as the default choice for hallucination detection.

Streszczenie

Zjawisko halucynacji dużych modeli językowych (*ang. large language models, LLMs*) stanowi istotną przeszkodę w ich wdrożeniu w wielu dziedzinach, gdzie potencjalny błąd może prowadzić do poważnych konsekwencji, np. w medycynie lub obszarze porad prawnych. Zaproponowanych zostało wiele metod do detekcji halucynacji w odpowiedziach modelu. Niniejsza praca porównuje dwa główne podejścia do wykrywania halucynacji na podstawie wewnętrznego stanu modelu: metody bazujące na stanach ukrytych oraz metody bazujące na mapach uwagi. Badania przeprowadzono w różnych scenariuszach eksperymentalnych, obejmujących różne zadania, zbiory danych, strategie dzielenia odpowiedzi na fragmenty oraz ustawienia hiperparametrów. Wyniki pokazują, że żadne z tych podejść nie osiąga przewagi we wszystkich możliwych scenariuszach, co podkreśla potrzebę dostosowania metod wykrywania halucynacji do specyfiki danego zadania. Metody oparte na stanach ukrytych lepiej sprawdzają się przy ocenie całych odpowiedzi oraz w zadaniu odpowiadania na pytanie. Z kolei metody bazujące na mapach uwagi są skuteczniejsze w zadaniu streszczania tekstu oraz przy analizie odpowiedzi podzielonych na fragmenty. Biorąc pod uwagę powyższe, metodą która cechuje się największą uniwersalnością jest *AggTruth* (bazująca na mapach atencji) i jest polecana jako podstawowe narzędzie do wykrywania halucynacji.

Contents

1. Introduction

The release of the transformer architecture [42] has been a milestone event in the domain of natural language processing (NLP). Since then, many transformer-based large language models (LLMs) have emerged, named after the enormous number of their parameters. Examples include ChatGPT [34], LLaMA [41], Mistral [21], and Gemma [32]. LLMs perform notably well in various NLP tasks [5, 23, 24, 25], yet questions arise about their truthfulness.

1.1. Problem Statement

LLMs tend to hallucinate [6, 44, 49]. While definitions of hallucination may vary across specific NLP tasks, the one most broadly accepted refers to it as *the generated content that is nonsensical or unfaithful to the provided input* [13, 31]. This property of LLMs hinders their application in real-world high-stakes scenarios such as medical advice, financial operations, or legal services [43].

One of the techniques that can help improve the truthfulness of LLMs is retrieval-augmented generation (RAG) [28]. It relies on enriching the input with extra documents (context) and instructing the model to answer based solely on them. However, even with RAG employed, models still encounter difficulties generating factual content [8]. Those problems emerge from inability of the model to extract relevant facts from context that contains noise and counter-facts or requires integrating information from different fragments. Furthermore, models should generate an appropriate response when the provided context is not sufficient. In this work, the author refers to hallucinations that arise from the model answering based on some provided context as *contextual hallucinations*.

Among various approaches aimed at detecting hallucinations, one can distinguish hidden state-based and attention map-based methods. Hidden state-based ones stem from research revealing that factual knowledge is stored in the parameters of the feed-forward network of transformers [11]. Attention map-based ones rely on the observation that attention heads are excessively involved in the transfer of knowledge, as they are catalysts for interactions between tokens [14, 48]. In both cases, an appropriate part of the model's internal state is saved during output generation and serves as the basis for constructing input features for a hallucination detector.

The main question to be answered in this work is: **Which of the two, hidden states or attention maps, are more suitable as features for detecting hallucinated responses?**

1.2. Thesis Objectives

The present work represents the first step toward constructing an online hallucination mitigation method based on guided decoding, with the hallucination detector being a crucial

component. To support further advancements in this area, the following research questions were examined in the context of various hallucination detection methods:

- **RQ1:** How does the choice of specific layers and/or attention heads influence the performance of the hallucination detector?
- **RQ2:** What are the differences in detection performance between evaluating the entire generated answer versus evaluating fixed-length k-token fragments? The latter more closely simulates online hallucination detection during response generation.
- **RQ3:** How does the hallucination detector perform in question answering tasks compared to text summarization tasks?
- **RQ4:** Is there a single method that achieves consistently accurate and stable results across all examined tasks, models, and datasets?

1.3. Thesis Outline

- **Chapter 1** introduces the problem addressed in this thesis and defines the granular research objectives in the form of *research questions*.
- **Chapter 2** presents a review of the literature related to hallucinations in large language models, describing existing approaches to their detection.
- **Chapter 3** provides a description of the transformer architecture, introducing key concepts such as the attention maps and hidden states.
- **Chapter 4** describes the methods selected for comparison, with particular attention to the way features are constructed for the hallucination detector.
- **Chapter 5** outlines the experimental setup shared by all experiments and subsequently presents individual experiments along with an analysis of their results.
- **Chapter 6** concludes the thesis by summarizing the main findings and suggesting potential directions for future research.

1.4. Contributions

This thesis builds upon research related to a novel method for detecting contextual hallucinations, referred to as *AggTruth*, originally presented in [30] and co-authored by the author of this master's thesis. The author's contributions to this work include, but are not limited to: proposing one of the four attention aggregation methods (*CosSim*, described in a later section), conducting an in-depth analysis of a key competing approach and comparing its performance against *AggTruth*, preparing the labeled datasets used to train the hallucination detection model and performing the experiments. This thesis extends the cited work by providing a thorough evaluation of the *AggTruth* method on previously unexplored scenarios and comparing it to competing methods it has not been compared with before.

2. Related Work

Modern natural language processing (NLP) has been revolutionized by the rise of Large Language Models (LLMs) built on the Transformer architecture [42]. These models operate by predicting the next token in a sentence, where a token is a small unit of text such as a word or subword. Transformer uses a self-attention mechanism to interpret each token in relation to its surrounding context, allowing the model to capture complex dependencies in language. By training on massive text corpora, LLMs learn statistical patterns in language and develop internal representations that encode broad world knowledge. The Transformer architecture is described in detail in Chapter 3.

Building on this foundation, models like LLaMA [41] have scaled up to hundreds of billions of parameters and are applied across a wide range of tasks, including translation, summarization, question answering, and code generation [5, 23]. Their effectiveness comes from the ability to perform well on new tasks simply by conditioning on carefully designed prompts, without requiring additional task-specific training. Despite these capabilities, LLMs have important limitations. Because they rely on statistical associations in text rather than true understanding or logical reasoning, they can sometimes generate outputs that appear fluent and convincing but are in fact incorrect or ungrounded in reality. This phenomenon is commonly referred to as *hallucination*.

2.1. Hallucinations

The hallucination phenomenon has been widely studied and a variety of taxonomies have been proposed. Most generally, hallucinations can be divided into intrinsic and extrinsic [20]. The former occur when the generated output directly contradicts the source content (i.e. user input), while the latter involve output that cannot be verified from the source content. On the other hand, a novel classification was proposed [18], which distinguishes factuality hallucination and faithfulness hallucination. Factuality hallucination refers to outputs that are either in conflict with real-world grounded facts or are unverifiable. Faithfulness hallucination pertain to situations when the model output is inconsistent with user's instruction, unfaithful with the contextual information provided by the user or exhibits internal logical conflict. Recent work [37] introduced an additional dimension to hallucination analysis by incorporating severity levels, referred to as degree. The levels of *alarming*, *moderate* and *mild*, represent a gradation in the potential seriousness of hallucinations, ranging from serious factual errors to minor and less harmful mistakes. These works highlight diverse taxonomies that extend beyond the traditional intrinsic/extrinsic distinction for describing hallucinations.

The work [44] addresses the hallucination problem from the perspective of learning theory. The authors define a formal world and demonstrate that LLMs cannot be used as general problem solvers and will hallucinate in this setting. Since the formal world represents only a subset of the real one, the occurrence of hallucinations cannot be fully avoided.

2.2. Retrieval-augmented generation

Retrieval-Augmented Generation (RAG) is an approach designed to ground LLM outputs in external knowledge and thereby reduce hallucinations [28]. In a typical RAG setup, an LLM is paired with a retrieval module that, given a user query, fetches relevant documents from an external corpus. The language model then conditions its response on both the original input and the retrieved content. This method has been shown to significantly improve performance on knowledge-intensive tasks by providing factual grounding [47].

However the quality of the final output depends heavily on the relevance and accuracy of the retrieved documents. If the retriever returns irrelevant, outdated, or noisy content, LLM may still hallucinate. For example, irrelevant passages can reduce answer accuracy in open-domain QA [46]. More in-depth analysis [8] showed that RAG systems are prone to several failure modes: noise robustness (difficulty identifying relevant information when documents are topically related but do not contain the answer), negative rejection (inability to abstain from answering when no retrieved document contains the necessary information), information integration (struggles with combining evidence from multiple sources to answer complex queries), and counterfactual robustness (susceptibility to reproducing known factual errors). Therefore, while RAG improves factual grounding, it does not completely solve hallucination problems, which still demand further mitigation efforts.

2.3. Sampling-based approaches

Sampling-based approaches aim to detect hallucinations by examining the consistency among multiple model-generated responses. This idea was extensively explored in [29], where the authors proposed SelfCheckGPT — a zero-resource black-box method for evaluating model outputs. It is based on intuition that when the model has the knowledge required to perform a task, then sampled responses should be similar to each other and they should differ otherwise. Authors proposed five different techniques for assessing the similarity of sampled responses.

The idea of comparing different responses was further developed in [12], but this time using the internal state of the model. In addition, instead of sampling multiple responses for the exact same prompt, an ancillary model is used to generate various and detailed questions based on the prompt and responses for these questions are assessed. Authors formalize their idea by introducing a new metric called *semantic entropy*, entropy calculated over meanings of sentences.

The main flaw of these black-box methods is the need of fully generating many responses. Not only does it require time and resources but also precludes detecting hallucinations in real-time, as generated responses can only be evaluated post-hoc. Consequently, these methods cannot be utilized in the on-line manner during the decoding process in order to influence the selection of tokens in favor of more factual ones. This limitation motivates the development of more efficient, real-time hallucination detection techniques that can intervene during the generation process.

2.4. Hidden state-based approaches

In order to detect hallucinations either generated output or internal state of the model can be examined. Generally, internal state refers to attention maps and outputs of feed-forward networks in the subsequent layers of the transformer architecture. The latter are also known as hidden states. According to research [3], a multi-layer perceptron classifier is able to assess truthfulness of a statement based on hidden states recorded while feeding the model with this statement (*SAPLMA* method). The key hypothesis is that hidden states encode enough information to distinguish truthful from false statements. Although models may generate incorrect but confident responses, their internal states may implicitly *know* when outputs are untruthful. However, this study was limited to simple true-false tasks without testing more complex, open-ended scenarios.

INSIDE method [7] is a more complex approach, that utilizes eigenvalues of covariance matrix of sampled responses embeddings, as they capture correlation between these responses. This approach can be treated as white-box alternative to SelfCheckGPT [29], however it likewise suffers from necessity to sample many responses and additionally requires an ancillary model to calculate the embeddings.

Meanwhile, another work [26] introduced a method to approximate semantic entropy using only hidden states from a single generation, called *Semantic Entropy Probes*. This addresses the main limitation of the method proposed in [12], which requires generating multiple samples per query. This technique estimates semantic entropy without access to token probability distributions during generation and employs a logistic regression model trained to assess semantic entropy levels for hallucination detection. SAPLMA and Semantic Entropy Probes are described in detail in Chapter 4.

2.5. Attention map-based approaches

Attention maps can be analyzed instead of hidden states in order to create meaningful features for hallucination detector. *Lookback-Lens* (described in detail in Chapter 4) was the first attempt to detect contextual hallucinations only using attention maps [9]. The idea is that when a language model begins to hallucinate, it increasingly relies on its previously generated tokens rather than the input context provided by the user. The method is based on quantifying the extent to which attention is put on the provided context versus on the newly generated content. An interesting property of the proposed solution is its transferability: a classifier trained on a 7B parameter model was shown to generalize well to a 13B model without requiring retraining. Additionally, the authors presented a practical example of employing the detector during the decoding process, resulting in improved factuality of the responses.

The LapEigvals method, proposed in [4], is based on interpreting attention maps as adjacency matrices of graphs, enabling the application of graph theory tools for hallucination detection. Attention maps are interpreted as adjacency matrices of directed graphs, where nodes correspond to tokens and edges represent the attention scores between them. For each attention map, the corresponding Laplacian matrix is computed, and its eigenvalues are extracted. The largest eigenvalues are then used as input features for a classifier trained to detect hallucinations.

2.6. Additional considerations

The results of [35] demonstrate that specific tokens have more meaningful influence on performance of a hallucination detector than others, however an automated method for recognizing these factual tokens remains a challenge. Besides, the investigation suggests that such detectors generalize poorly across datasets. The authors of [10] propose a novel decoding technique that improves factuality based on contrasting outputs of the last layer of the model with another dynamically selected one. Although it is not a hallucination detection method as such, the work empirically proves that findings from previous research about transformer-based language models storing syntactic information at earlier layers and semantic information higher in the network [15], are related to the hallucination problem. It has been shown that LLMs are capable of efficiently assessing the truthfulness of their own generations [22], suggesting that they possess some internal representation of truthfulness. Similarly, the authors of [2] collected a dataset of *Known-Unknown* questions and fine-tuned models to distinguish between answerable and unanswerable queries. Recent study by [39] showed that models seem to encode (un)answerability of the input query. This internal encoding suggests that models possess latent awareness of whether a question can be answered, even if their generated responses do not always reflect this. Exploiting this insight could lead to improved hallucination detection by identifying when a model's confidence is misplaced.

2.7. Summary

The phenomenon of hallucination in LLMs has been extensively explored, with multiple taxonomies proposed to classify its forms. These taxonomies include distinctions such as intrinsic vs. extrinsic hallucinations, factuality vs. faithfulness hallucinations, and severity-based gradations, highlighting the complex nature of hallucinations. Retrieval-Augmented Generation (RAG) has emerged as a prominent method for hallucination mitigation by grounding LLM outputs in external, retrieved knowledge sources. While RAG significantly improves factual grounding on knowledge-intensive tasks, its effectiveness depends heavily on the relevance and accuracy of retrieved documents. This highlights the necessity for hallucination detection methods, among which a variety of approaches can be identified. Sampling-based methods, such as SelfCheckGPT, rely on the diversity of multiple model outputs to infer factuality. These approaches, while intuitive and effective, are computationally intensive and unsuitable for real-time use. Analyzing the internal state of the model is an alternative. Research shows that hidden state of a properly selected token can implicitly contain signals about (un)truthfulness, no matter whether the actually generated response is factual or not. Attention map-based methods analyze attention distributions to infer hallucinations. Techniques like Lookback-Lens detect when models rely increasingly on generated content rather than the input, offering efficient, transferable detectors that can operate during decoding. Graph-based methods like LapEigvals further extract structural patterns from attention maps for classification. Additional considerations highlight limitations and future directions, such as the importance of specific *factual* tokens and the poor generalization of detectors across datasets, as well as the potential for leveraging the model's latent awareness of answerability to enhance hallucination detection strategies.

3. Transformer

Modern large language models rely on the transformer architecture [42], originally designed for the task of translating between natural languages (machine translation). A variant of this architecture, called the decoder-only transformer, is capable of generating text in an autoregressive manner. It means that given a sequence of input tokens, it is able to predict the next one by conditioning on the provided ones. This chapter describes the architecture of the decoder-only transformer, introducing the concepts of hidden states and attention maps.

3.1. Overview

A core concept in the transformer architecture is a *token*. It is a piece of text, usually a word or subword, that serves as the basic unit of input and output for the model. The model generates text by predicting tokens one by one in what is known as an *autoregressive* or *causal* manner, where each token is predicted based on the preceding ones. Having predicted a token, it is added to the end of the previously generated tokens, and the model generates the next token based on the updated sequence. This continues until a special end-of-sequence token is produced or a predefined maximum length is reached. Fig. 3.1 sketches the whole process.

Tokens are first transformed into embeddings, which are high-dimensional vectors that capture the semantic meaning of each token. Since these embeddings lack positional information, positional encodings are added to incorporate the order of tokens in the sequence. The resulting position-aware embeddings are then passed to the transformer layers for further processing.

The embeddings are then processed in alternating layers of masked self-attention and feed-forward neural networks. Attention allows tokens to interact with one another, with the important constraint that each token can only attend to itself and the tokens that precede it and not the ones that follow. This is what makes the model causal and suitable for autoregressive generation. The attention mechanism enriches each embedding by incorporating information from previous tokens, effectively refining its meaning based on context. As a result, the vectors at this stage are no longer the original token embeddings from the vocabulary. They have been transformed into contextual embeddings that reflect both the token itself and its surrounding context.

After the masked self-attention layer, each token representation is passed through a feed forward neural network (FFN). Unlike the attention mechanism, which allows tokens to interact with each other, the FFN operates independently on each token, applying the same transformation to all positions. Although the FFN does not introduce new contextual information, it can be thought of as a kind of learned key-value lookup table, where different semantic patterns are encoded in the weights of the network. By applying this transformation, the FFN enriches the token embeddings with higher-level semantic features that go beyond their original meaning.

The final step is the selection of the next token. A linear layer is fed with the enriched embedding of the most recently generated token. This layer produces a vector whose length is equal to the size of the vocabulary, where each value represents the model's confidence that the corresponding token should be the next one. Based on this vector, the next token is selected using a decoding strategy, which may be greedy selection (choosing the token with the highest confidence) or a probabilistic sampling method.

3.2. Input Encoding

Tokens are first transformed into embedding vectors, which are real-valued vectors representing each token as a point in a continuous, high-dimensional space. The size of these vectors is referred to as the *hidden dimension*. These embeddings capture semantic relationships, as tokens with similar meanings tend to be mapped to nearby points. This transformation is performed via an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d_{\text{model}}}$, where V is the vocabulary size and d_{model} is the model's hidden dimension. Each token is mapped to a row vector in $\mathbf{E}$.

However, these embeddings alone contain no information about token order. Since the Transformer processes all tokens in parallel and lacks any notion of space, positional information is critical. To address this, fixed or learned positional encodings are added element-wise to the token embeddings. This step injects information about the position of each token within the sequence, enabling the model to incorporate word order.

The final position-aware embeddings, combining semantic and positional information, serve as the input to the subsequent Transformer blocks.

3.3. Masked Self-Attention

Attention mechanism enables the model to determine the extent to which individual tokens influence each other during processing leading to enriched contextual embeddings. It is a crucial feature as the same tokens can have different meanings depending on the other tokens in the sequence. All attention calculations are performed in parallel across different *heads*. Each head is a set of neural networks with identical architecture but different learned weights. This design allows individual heads to specialize in capturing specific types of relationships between tokens, which enhances the model's expressiveness and overall performance.

Given an input vector $\mathbf{x}_i \in \mathbb{R}^{d_{\text{model}}}$, which is the embedding of a token, the attention mechanism produces an output vector $\mathbf{a}_i$ that corresponds to $\mathbf{x}_i$ but is enriched with contextual information from other tokens. To compute $\mathbf{a}_i$, the mechanism first derives three distinct representations: the *query*, *key*, and *value* vectors. These are obtained by multiplying the input with learned projection matrices (unique for each head):

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q, \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^K, \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^V, \tag{3.1}$$

where the weight matrices have the following shapes:

$$\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad \mathbf{W}^V \in \mathbb{R}^{d_{\text{model}} \times d_v}.$$

The *attention score* between the i-th token and the j-th token is calculated as the dot product of their corresponding query and key vectors, scaled by the square root of the query/key dimensionality:

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}. \tag{3.2}$$

The scaling is introduced to mitigate numerical instability and prevent potential gradient explosion, which may result from large values of the dot product.

The properties of matrix multiplication enable parallel computation of attention scores between all tokens in the sequence. This results in a matrix $\boldsymbol{\alpha} \in \mathbb{R}^{n \times n}$, also known as the **attention map**, where n is the number of tokens seen so far. Each row $\boldsymbol{\alpha}_i$ is obtained by applying the *softmax* function to the vector of scores between token i and all previous tokens $j \leq i$:

$$\boldsymbol{\alpha}_i = \text{softmax}\Big(\{\text{score}(\mathbf{x}_i, \mathbf{x}_j)\}_{j \leq i}\Big). \tag{3.3}$$

Each element α_{ij} of this vector indicates how much token j contributes to enriching the representation of token i.

The *softmax* function takes a vector of real-valued scores and transforms it into a probability distribution. It does this by exponentiating each element to ensure all values are positive, and then normalizing by the sum of all exponentiated values. Formally, for a vector $\mathbf{z} = (z_1, z_2, \ldots, z_m)$, *softmax* is defined as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{k=1}^{m} \exp(z_k)} \quad \text{for } i = 1, \ldots, m. \tag{3.4}$$

This transformation preserves the relative magnitudes of the scores—higher scores result in higher probabilities—but ensures that the resulting values sum to 1, making them interpretable as probabilities. In the context of attention, applying *softmax* to the scores between tokens produces a distribution over those tokens, indicating the proportion (or weight) with which each token should contribute to enriching the representation of the current token.

To preserve the autoregressive property of the model, a causal mask is applied before *softmax* to prevent tokens from attending to future positions. This masking is precisely why the mechanism is called *masked* self-attention. This implies that the attention map is a square lower-triangular matrix, which grows by one row and one column with each generation step.

Having computed the attention scores between tokens, the output of a single attention head for token i is calculated as a weighted sum of the value vectors of tokens $j \leq i$:

$$\mathbf{head}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j. \tag{3.5}$$

The outputs from different heads are ultimately concatenated and multiplied by a weight matrix $\mathbf{W}^O$, which ensures that the resulting vector $\mathbf{a}_i$ has the same dimensionality as the input embedding $\mathbf{x}_i$:

$$\mathbf{a}_i = \text{Concat}(\mathbf{head}_i^{(1)}, \mathbf{head}_i^{(2)}, \ldots, \mathbf{head}_i^{(H)})\mathbf{W}^O, \tag{3.6}$$

where H is the number of attention heads.

3.4. Feed Forward Network

In addition to the attention sub-layers, each layer of the decoder contains a fully connected feed-forward network (FFN), which is applied independently and identically to each token position. This network consists of two linear transformations separated by a ReLU activation function:

$$\mathbf{h}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{3.7}$$

where the weight matrices and bias vectors have shapes:

$$\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}, \quad \mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}, \quad \mathbf{b}_1 \in \mathbb{R}^{d_{\text{ff}}}, \quad \mathbf{b}_2 \in \mathbb{R}^{d_{\text{model}}}.$$

The output of the FFN, denoted as $\mathbf{h}(\mathbf{x})$, represents the updated **hidden state** of the token position within the current layer. The hidden state is a vector embedding that captures the contextualized representation of a token at that specific layer, incorporating information aggregated from attention and feed-forward operations.

Although the same FFN parameters are shared across all positions within a layer, each layer has its own distinct parameters. Typically, the inner-layer dimensionality d_{ff} is larger than the model dimensionality d_{model}. This expansion allows the network to learn richer and more expressive representations. The ReLU activation function introduces non-linearity into the network, enabling it to model complex patterns and interactions between tokens.

3.5. Language Modeling Head

The ultimate goal of transformer is to predict the next token. To do this, the hidden state of the most recently generated token from the last layer, denoted $\mathbf{h}_N^L$, is passed through a linear layer:

$$\mathbf{z} = \mathbf{h}_N^L \mathbf{E}^\top, \tag{3.8}$$

where $\mathbf{E} \in \mathbb{R}^{V \times d_{\text{model}}}$ is the embedding matrix introduced earlier. The weights of this linear layer can be learned independently, but it is common practice to tie them to the transpose of the embedding matrix $\mathbf{E}$. During training, the matrix $\mathbf{E}$ is optimized to serve both as the embedding lookup and as the output projection, effectively working in both directions. The resulting vector $\mathbf{z} \in \mathbb{R}^V$ is known as the **logits**, which represent unnormalized scores for each token in the vocabulary.

Finally, to obtain a probability distribution over the vocabulary, a *softmax* function with an optional *temperature* parameter $\tau > 0$ is applied to $\mathbf{z}$:

$$P(k \mid \mathbf{z}) = \frac{\exp(z_k/\tau)}{\sum_{v=1}^{V} \exp(z_v/\tau)}, \tag{3.9}$$

where z_k is the k-th element of $\mathbf{z}$, corresponding to the score (logit) for the k-th token in the vocabulary of size V. This formula computes the probability that token k will be the next token generated by the model. The temperature τ controls the sharpness of the distribution: lower values produce more confident predictions (in particular, leading to always sampling the most probable token), while higher values yield softer distributions, allowing for greater creativity in the model's generations.

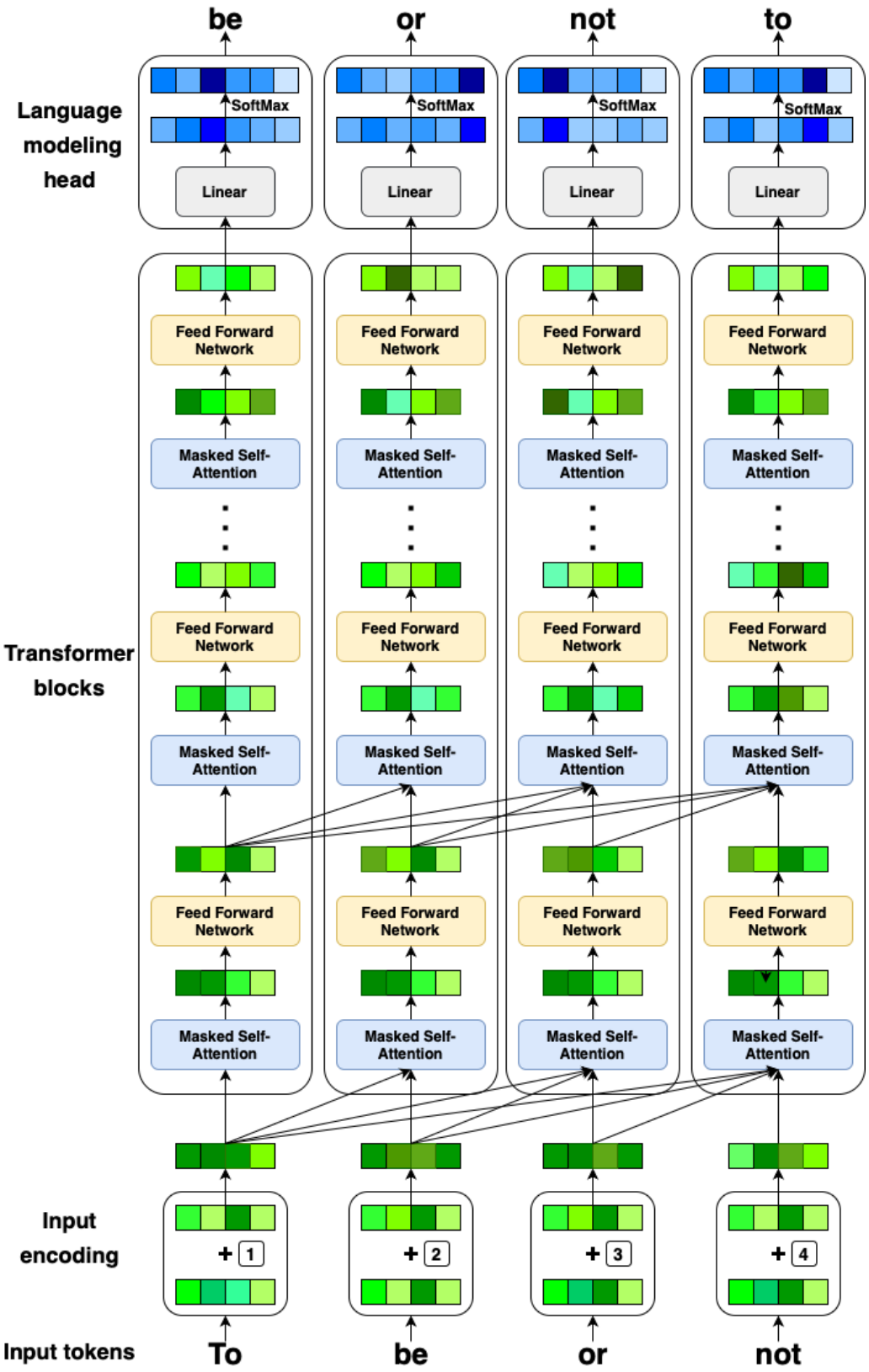

Figure 3.1: The process of generating tokens by transformer.

4. Hallucination Detection Methods

This chapter presents various hallucination detection methods based on hidden states and attention maps. The selected methods were chosen according to their suitability for the intended experimental scenarios. For instance, the INSIDE method was excluded because it requires sampling multiple responses at inference time, which is incompatible with the setup. The experimental scenarios were designed with the goal of identifying the method best suited for integration into an online hallucination mitigation pipeline based on guided decoding, where a reliable hallucination detector serves as a crucial component. The notation used throughout the chapter is introduced at the beginning.

Let M be a large language model that generates a sequence of tokens $(x_1, x_2, \ldots, x_T)$. For each token x_t, the model computes a hidden state vector at layer l, denoted by $\mathbf{h}_t^{(l)} \in \mathbb{R}^d$, where d is the dimensionality of the hidden representation. These hidden states collectively form a three-dimensional matrix of shape $L \times T \times d$, where L is the number of layers and T is the number of generated tokens.

Simultaneously, an attention matrix of shape $L \times H \times T \times S$ can be extracted from the model, where H denotes the number of attention heads and S is the length of the full input sequence, consisting of both the prompt tokens provided by the user and the tokens generated so far. This tensor captures the attention weights assigned to each token in the sequence across layers and heads. Let $a_{t,i}^{(l,h)}$ denote the attention score assigned by head $h \in \{1, \ldots, H\}$ at layer $l \in \{1, \ldots, L\}$, for the generated token x_t, where $t \in \{1, \ldots, T\}$, attending to the token at position $i \in \{1, \ldots, S\}$. The vector $a_t^{(l,h)} \in \mathbb{R}^S$ represents the full attention distribution over all tokens in the sequence at generation step t, corresponding to layer l and head h.

All the methods presented in this chapter use a logistic regression model to build a classifier in order to ensure comparability across methods. It was selected due to its simplicity, interpretability and effectiveness. It allows for a direct interpretation of feature importance, which aids in understanding model behavior. Logistic regression models the probability of assigning label 1 (i.e., detecting a hallucination) based on a vector of input features. Formally, it can be described as:

$$\hat{y} = \sigma\left(\boldsymbol{\beta}^\top \mathbf{z} + \beta_0\right), \tag{4.1}$$

where: $\hat{y}$ – the predicted probability of the label being 1, σ – the sigmoid function, which maps the input to the range $[0, 1]$, $\boldsymbol{\beta}$ – the weight vector (model parameters) learned during training, β_0 – the bias term (scalar), $\mathbf{z}$ – the vector of input features.

The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

4.1. SAPLMA

SAPLMA (Statement Accuracy Prediction based on Language Model Activations) is a hidden state-based method that stems directly from the hypothesis that the hidden states of an LLM contain information about whether the model estimates a sentence to be true or false [3]. It is specifically tailored for detecting factual hallucinations, as defined in the taxonomy proposed in [18].

This simple method involves training a classifier using the hidden states of an LLM. The task of the classifier is to determine whether the generated sequence of tokens is truthful or hallucinated (i.e., contains at least one untruthful token). The hidden state value for the last generated token (the one preceding the end-of-sequence token) is examined, and the choice of layer is arbitrary. The authors proposed employing a shallow neural network as the classifier. However, in this work, it is replaced with a logistic regression model (see Eq. 4.1) to allow a fair comparison with other methods that also utilize it.

4.2. Semantic Entropy Probes

Semantic Entropy Probes (SEPs) were proposed in [26] as a computationally efficient hidden state-based extension of the semantic entropy-based hallucinaton detector [12]. In both studies, hallucination detectors utilize the value of semantic entropy (SE), a measure that describes how certain the model is about the answer it generates. The main drawback of the original approach is the necessity of generating the response multiple times, typically 10, in order to compute SE. This results in a significant and often prohibitive increase in both time and resource consumption. SEPs make it possible to predict SE using only the hidden states of a single generation.

The process of calculating SE for a query consists of three steps: (1) sampling N responses at high temperature ($\tau = 1$) for a query q, (2) grouping the responses into clusters of semantic equivalence, and (3) estimating cluster probabilities and calculating SE. Step (1) is self-explanatory. Step (2) requires determining whether two generations have the same meaning (semantic equivalence). The authors propose utilizing either a strong LLM, such as GPT-4o, or a natural language inference (NLI) model, such as DeBERTa [17], for this task. In this work, the second option was chosen. In this setting, two generations are considered to have the same meaning if at least one entails the other and neither contradicts the other. Clusters are then created using the following algorithm: for each sample s_a, it is either added to an existing cluster C_k if semantic equivalence holds between s_a and a sample $s_b \in C_k$, or a new cluster is created if sample s_a has a different meaning than all current clusters. This step results in the samples being divided into K clusters. Step (3) involves estimating cluster probabilities $p(C_k \mid q)$ as the fraction of samples that fall into that cluster:

$$p(C_k \mid q) = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}[s_j \in C_k] \tag{4.3}$$

Semantic entropy is then calculated as the entropy of the categorical distribution:

$$H_{\mathrm{SE}}(q) := -\sum_{k=1}^{K} p(C_k \mid q) \log p(C_k \mid q) \tag{4.4}$$

Once SE is calculated, the dataset consisting of $\{\mathbf{h}^{(l)}(q_j), \tilde{H}_{\mathrm{SE}}(q_j)\}_{j=1}^{Q}$ pairs is constructed, where $\mathbf{h}^{(l)}(q) \in \mathbb{R}^d$ denotes the hidden state vector corresponding to the last generated token (i.e., the token preceding the end-of-sequence token) and $\tilde{H}_{\mathrm{SE}}(q) = \mathbb{1}[H_{\mathrm{SE}}(q) > \gamma^\star]$ is a binarized version of $H_{\mathrm{SE}}(q)$ based on the threshold $\gamma^\star$, which is selected by minimizing the intra-class variance defined by the following equation:

$$\gamma^\star = \arg\min_{\gamma} \sum_{j \in \mathrm{SE}_{\mathrm{low}}} \left(H_{\mathrm{SE}}(q_j) - \hat{H}_{\mathrm{low}}\right)^2 + \sum_{j \in \mathrm{SE}_{\mathrm{high}}} \left(H_{\mathrm{SE}}(q_j) - \hat{H}_{\mathrm{high}}\right)^2 \tag{4.5}$$

where

$$\mathrm{SE}_{\mathrm{low}} = \{j : H_{\mathrm{SE}}(q_j) < \gamma\}, \qquad \mathrm{SE}_{\mathrm{high}} = \{j : H_{\mathrm{SE}}(q_j) \geq \gamma\},$$

$$\hat{H}_{\mathrm{low}} = \frac{1}{|\mathrm{SE}_{\mathrm{low}}|} \sum_{j \in \mathrm{SE}_{\mathrm{low}}} H_{\mathrm{SE}}(q_j), \qquad \hat{H}_{\mathrm{high}} = \frac{1}{|\mathrm{SE}_{\mathrm{high}}|} \sum_{j \in \mathrm{SE}_{\mathrm{high}}} H_{\mathrm{SE}}(q_j).$$

The authors propose to binarize the target variable, since the ultimate goal is to detect whether a response is hallucinated or not. Nevertheless, the logistic regression model outputs a probability, which allows retaining more nuanced information even after converting the task into a binary classification problem.

Having assembled the dataset, the logistic regression model (see Eq. 4.1) can be trained to predict the SE level. Once trained, this classifier may be employed to assess the truthfulness of responses. According to the authors, this approach remains effective despite the model being optimized for a different task, owing to the correlation between SE and hallucination probability. A notable limitation of the SEPs method is its inapplicability in scenarios involving windowed responses, since only SE values computed for entire responses are available for training. On the other hand, this method does not require a labeled dataset, as SE is calculated based on extra sampled responses.

4.3. Lookback-Lens

Lookback-Lens, proposed in [9], offers a novel perspective on the problem of hallucination detection by relying exclusively on attention maps. What distinguishes this method is its design, which is specifically tailored to detect **contextual** hallucinations. It is founded on the idea that such hallucinations occur when the model fails to pay enough attention to the provided input.

At the core of Lookback-Lens is a metric called the *lookback-ratio*. Given an input sequence of prompt tokens $(p_1, p_2, \ldots, p_P)$ of length P, and a sequence of previously generated tokens $(x_1, x_2, \ldots, x_{t-1})$, the lookback-ratio for a newly generated token x_t is defined as:

$$\mathrm{LR}_t^{(l,h)} = \frac{A_t^{(l,h)}(\text{prompt})}{A_t^{(l,h)}(\text{prompt}) + A_t^{(l,h)}(\text{new})} \tag{4.6}$$

where:

$$A_t^{(l,h)}(\text{prompt}) = \frac{1}{P}\sum_{i=1}^{P} a_{t,i}^{(l,h)}, \qquad A_t^{(l,h)}(\text{new}) = \frac{1}{t-1}\sum_{i=P+1}^{P+t-1} a_{t,i}^{(l,h)}.$$

For a given token x_t, the lookback ratios from all attention heads are concatenated into a vector

$$\mathbf{LR}_t = [\mathrm{LR}_t^{(1,1)}, \mathrm{LR}_t^{(1,2)}, \ldots, \mathrm{LR}_t^{(L,H)}].$$

The values for corresponding heads are then averaged across the considered window, resulting in a single vector $\mathbf{LR}^{h,t}$ that summarizes the entire span. Each element of this vector corresponds to a specific attention head in the transformer model. A logistic regression model (see Eq. 4.1) is then trained using $\mathbf{LR}^{h,t}$ as the feature vector for the task of hallucination detection. This method was designed to work with windowed responses, but in the case of no windowing scenario the whole response can be treated as a single window.

4.4. AggTruth (Proposed)

AggTruth is an attention map-based method for hallucination detection, specifically designed to address **contextual** hallucinations in the setting of **windowed** responses. Recall that for each generated response, the model produces a four-dimensional attention matrix of shape $L \times H \times T \times S$, where L is the number of layers, H is the number of attention heads, T is the number of generated tokens, and S is the total length of the input sequence (including both the user-provided prompt tokens and the generated tokens). AggTruth focuses only on a subset of this tensor, specifically a submatrix of shape $L \times H \times T \times C$, where C corresponds to the number of tokens in the provided context. This means that AggTruth discards attention information related to both the previously generated tokens and any parts of the prompt that fall outside of the explicitly defined context, such as system prompt or query. This design choice is motivated by the assumption that hallucinations are primarily influenced by improper utilization of the provided context. Including unrelated tokens, such as system instructions or distant prompt segments, may introduce noise and dilute the signal necessary for accurate detection.

The attention score vector of each generated token is subsequently aggregated using one of the four proposed aggregation techniques. Fig. 4.1 presents an example of attention map with context tokens outlined in red. Single value is produced out of every highlighted vector. The aggregation is necessary because, in order to train a hallucination detection model, the attention scores over the provided context for each generated token must be condensed to reduce both the dimensionality and the volume of the attention maps. Aggregation also helps distill the most relevant signal from the attention maps, enabling more robust and generalizable learning. The aggregated features are then averaged across the tokens within the context window, resulting in a single feature vector for each attention head. These final feature vectors are used as input to the hallucination detection model.

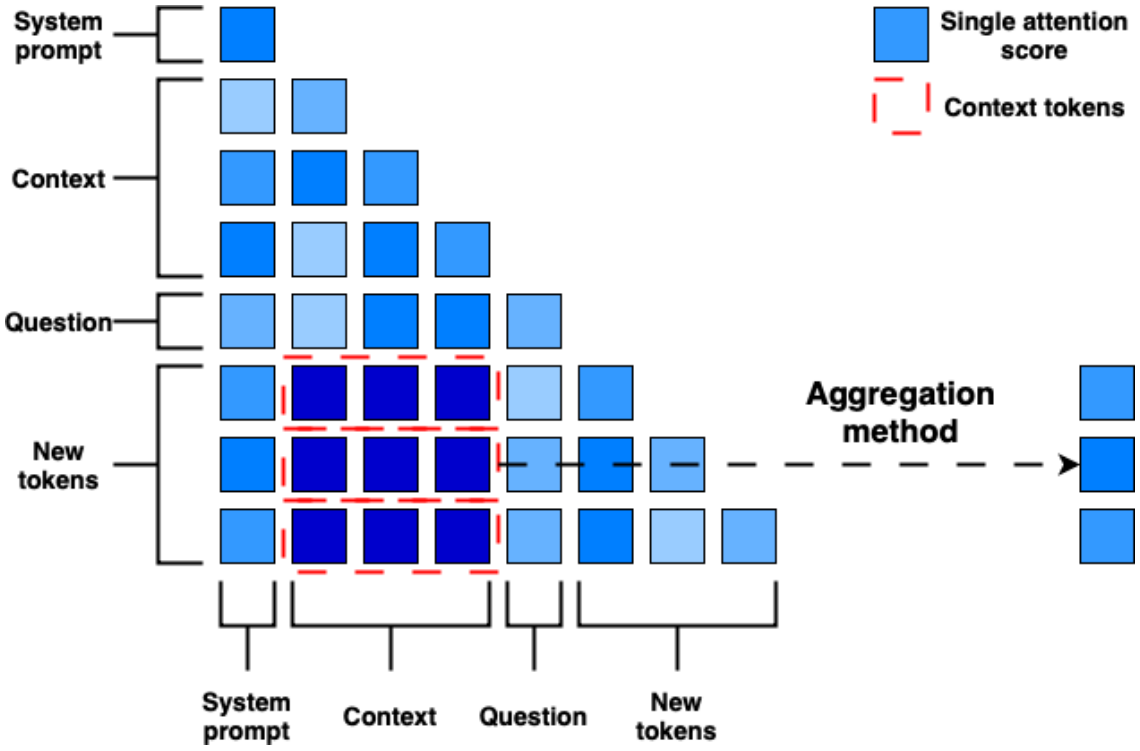

Figure 4.1: Example of attention map aggregation.

4.4.1. Attention aggregation techniques

AggTruth Sum

The most straightforward method to aggregate attention scores is by summing the attention weights assigned to the context tokens during the generation of each token, as shown in Equation 4.7.

$$\text{Sum}_t^{(l,h)} = \sum_{i=1}^{C} a_{t,i}^{(l,h)} \tag{4.7}$$

This method relies on the assumption that large language models produce more factual responses when they allocate greater attention to the provided context tokens. Since only tokens from the given context are considered, the sum of the attention scores is not constrained to equal 1. This property is beneficial because it allows differentiation between examples based on their total attention allocation.

AggTruth CosSim

Each head within a layer is represented by the attention distribution it assigns to the context tokens during the generation of a new token. The cosine similarity between the attention vectors of each pair of heads within the same layer is computed, and the values are averaged for each head, as formalized in Equation 4.8.

$$\text{CosSim}_t^{(l,h)} = \frac{1}{H-1} \sum_{\substack{h'=1 \\ h' \neq h}}^{H} \frac{\mathbf{a}_t^{(l,h)} \cdot \mathbf{a}_t^{(l,h')}}{\|\mathbf{a}_t^{(l,h)}\| \|\mathbf{a}_t^{(l,h')}\|} \tag{4.8}$$

This results in a single value per head, quantifying its similarity to other heads within the same layer when generating a particular token. The underlying intuition is that when the model produces unreliable content, the attention patterns across heads tend to converge, suggesting a lack of distinct focus.

AggTruth Entropy

Uncertainty is another way to understand hallucinations in LLM outputs. The entropy-based aggregation, defined in Equation 4.9, quantifies whether the model's attention is concentrated on specific tokens or more diffusely spread across the entire context during token generation.

$$\text{Entropy}_t^{(l,h)} = -\sum_{i=1}^{C} a_{t,i}^{(l,h)} \log_2 a_{t,i}^{(l,h)} \tag{4.9}$$

To ensure the attention scores are treated as proper probability distributions summing to one, an additional component is appended to the attention vector, representing the difference between one and the sum of the original attention scores.

AggTruth JS-DIV

Hallucinations can be viewed as outliers, reflecting uncertainty or gaps in the model's knowledge. This motivates the hypothesis that hallucinated and non-hallucinated examples differ in the stability of their attention distributions across layers. Treating attention scores as probability distributions allows us to compute the average attention distribution of heads within each layer. The Jensen-Shannon distance is then employed to measure the divergence between each individual head's distribution and this average, as shown in Equation 4.10.

$$\text{JS-Div}_t^{(l,h)} = \sqrt{\frac{1}{2} \sum_{i=1}^{C} \left(a_{t,i}^{(l,h)} \ln \frac{a_{t,i}^{(l,h)}}{m_{t,i}^{(l,h)}} + a_{t,i}^{(l,\text{ref})} \ln \frac{a_{t,i}^{(l,\text{ref})}}{m_{t,i}^{(l,h)}} \right)} \tag{4.10}$$

where

$$a_{t,i}^{(l,\text{ref})} = \frac{1}{H} \sum_{h=1}^{H} a_{t,i}^{(l,h)}, \quad m_{t,i}^{(l,h)} = \frac{a_{t,i}^{(l,h)} + a_{t,i}^{(l,\text{ref})}}{2}.$$

Similarly to the previous aggregation technique, an additional component is appended to the attention vector to ensure the scores form a proper probability distribution summing to one.

4.4.2. Context percentage feature

As new tokens are generated, the total number of tokens requiring attention from the language model increases. Consequently, the proportion of attention allocated to the provided context decreases, as the focus becomes distributed over an expanding sequence of tokens. This phenomenon can negatively impact certain attention aggregation methods, such as AggTruth Sum, where aggregated values naturally diminish over time. To mitigate this issue, an additional feature is incorporated into the dataset, representing the ratio of the context length to the total input length at the moment each token is generated.

4.4.3. Proper token-to-features alignment

The attention map generated during the production of a new token reflects the influence of preceding tokens on that token's prediction. However, the token being generated does not influence the attention scores at the time of its creation. In other words, the attention map is the same regardless of the decoding strategy used, whether greedy decoding or probabilistic sampling. A token begins to influence subsequent attention only after it has been generated. Based on this observation, it is hypothesized that, in order to construct meaningful features for the token at step t, the attention vector produced during the generation of the token at step $t + 1$ should be used. If this alignment is not taken into account, each token is effectively described by features corresponding to its predecessor.

5. Experiments

This chapter focuses on the empirical evaluation of hallucination detection methods proposed in the previous part of the work. The experimental effort is driven by a set of research questions that aim to uncover the strengths, limitations, and generalizability of different detection approaches across multiple tasks and configurations. Subsequent parts of the chapter describe the experimental setup and present individual experiments designed to address each research question, along with detailed analysis and discussion of the results.

5.1. Research goals

The primary objective of the experimental part of this work is to address the research questions outlined in Section 1.2, which focus on various aspects of hallucination detection in language model outputs. These investigations serve as a foundational step toward developing an online hallucination mitigation approach based on guided decoding, where the hallucination detector plays a central role.

The experiments aim to explore **(RQ1)** how the choice of specific layers and attention heads affects detection performance, **(RQ2)** the differences between evaluating full generated responses versus fixed-length token fragments, **(RQ3)** how detection effectiveness varies between question answering and summarization tasks and **(RQ4)** whether a single method can consistently yield accurate and robust results across different tasks, models, and datasets.

The specific methods, tools, and experimental techniques used to address these questions are described in detail in the following sections, each corresponding to a particular research question or group of related experiments.

5.2. Experimental setup

An automated pipeline was constructed to enable efficient evaluation and comparison of various hallucination detection methods across different LLMs and datasets (see Fig. 5.1).

The pipeline begins with a prompt and proceeds through several stages to ultimately generate features used to train a hallucination detector. First, an LLM generates an answer, and its internal state is saved (i.e., hidden state values and attention maps). Prompt template used to query all the models is as follows:

```
You are a helpful assistant. Your job will be to answer questions accurately
based on the given context and not your internal knowledge. If you cannot
answer the question only based on the provided context, return the answer:
`I cannot answer this question based on the provided context.`.
Given the context `CONTEXT` and the query `QUERY` below, please provide an
answer `ANSWER` to the question.
```

```
`CONTEXT`: {context}

`QUERY`: {question}

`ANSWER`:
```

The model's answers are then evaluated by another LLM using an LLM-as-a-judge approach. GPT-4o [19] was selected for this task, as it demonstrates a high level of agreement with human annotators [9]. Based on GPT-4o's evaluation, the generated tokens are labeled as hallucinated (1) or non-hallucinated (0). Responses can optionally be windowed—that is, divided into overlapping chunks of size k, sliding token by token. A response (or window) is assigned a label of 1 if it contains any hallucinated token, and 0 otherwise. Then, each response (or window) is aligned with its corresponding internal state, and a feature vector is constructed based on the chosen hallucination detection method. This process results in a final data set that contains a feature vector for each response (or window) along with its associated hallucination label.

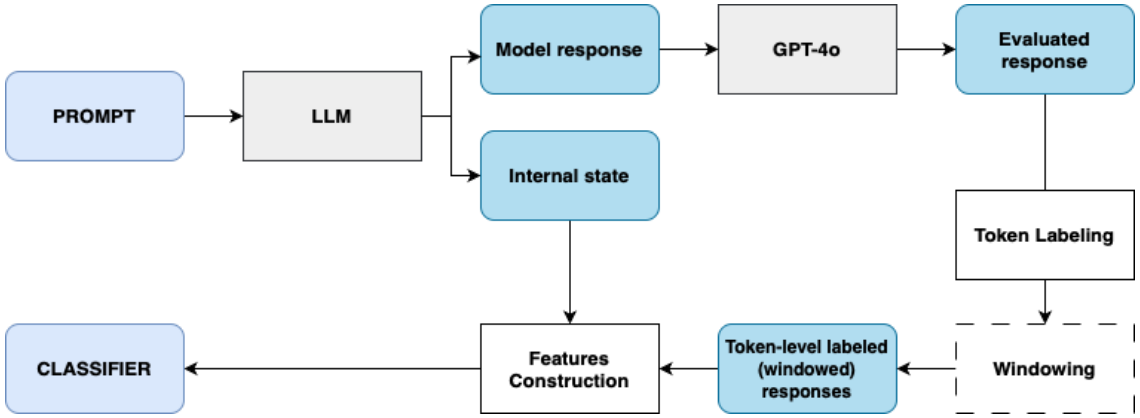

Figure 5.1: Automated pipeline used to benchmark hallucination detection techniques.

The resulting dataset is then used to train and evaluate the hallucination detection classifier. Specifically, a logistic regression model from the *scikit-learn* [36] library is employed, using the LBFGS solver and setting the hyperparameter C, which controls the strength of L2 regularization, to 0.1. Prior to classification, feature vectors are scaled using a MinMax scaler. This choice ensures that all input features contribute proportionally to the model, which is especially important given the use of L2 regularization, as it penalizes large coefficients and can be biased if features have vastly different scales. The results are reported using the Area under the ROC curve (AUC) metric. AUC represents the area under the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (TPR), the proportion of correctly identified positive samples, against the False Positive Rate (FPR), the proportion of negative samples incorrectly classified as positive, across all classification thresholds. It was chosen as it is a standard metric for binary classification tasks, offering a threshold-independent assessment of model performance. Its use is also well-established in the existing literature on hallucination detection.

5.2.1. Data

The experimental data originates from four different datasets: two related to Question Answering (QA) and two to Text Summarization (SUMM) task. The QA portion consists of 1819 examples from Natural Questions (NQ) [27] and 1050 from HotPotQA [45]. Each example contains a context passage and a question that should be answered based on that context. In some cases, the context lacks sufficient information to answer the question accurately. The SUMM portion includes 1000 examples from CNN/Daily Mail [38] and 1000 from XSUM [33]. Overall, the dataset consists of two distinct parts corresponding to different tasks. This choice was motivated by the desire to examine the generalization capabilities of the methods.

In real-world applications, the inputs to an LLM are likely to differ from the distribution of the training set used to develop the hallucination detector. To enable meaningful evaluation under such conditions, the data is structured as follows (see Tab. 5.1). For each task, one of the datasets is used for training, while the second is left out as the test set. Additionally, the performance of the detector is evaluated on the two datasets associated with the other task in order to assess the method's generalization.

Table 5.1: Data split per source/target task configuration.

Source	**Target**	**Source**		**Target**	
		Train/Val	**Same Task Test**	**Transfer (1) Test**	**Transfer (2) Test**
QA	SUMM	NQ	HotPotQA	CNN/DM	XSum
SUMM	QA	CNN/DM	XSum	NQ	HotPotQA

5.2.2. Models

Three different large language models were selected for experiments: LLaMA-3.1-8B [16], Phi-3.5-mini [1], and Gemma-2-9B (4-bit) [40]. In all cases, the *instruct* version was used. These models differ in hidden dimension and the number of layers and attention heads, providing a diverse experimental context. All of them have fewer than 10 billion parameters, allowing for deployment in resource-constrained scenarios. However, their relatively smaller size may also make them more susceptible to hallucinations. Tab. 5.2 compares the key parameters of the models under consideration and presents the percentage of non-hallucinated responses across all datasets. The responses generated by each model were produced using a greedy decoding strategy and subsequently evaluated by GPT-4o.

Table 5.2: Comparison of key models parameters and their performance during greedy decoding (the percentage of non-hallucinated responses).

LLM	**Layers**	**Heads**	**Hidden Size**	**NQ**	**HotPotQA**	**CNN/DM**	**XSum**
meta-llama/Llama-3.1-8B-Instruct	32	32	4096	72.3	55.0	86.1	78.0
microsoft/Phi-3.5-mini-instruct	32	32	3072	59.6	77.2	75.5	65.3
unsloth/gemma-2-9b-it-bnb-4bit	42	16	3584	83.3	85.5	86.3	78.0

5.3. Best obtained results

The following section presents the best AUC scores achieved by the examined methods for the hallucination detection task across all models and both source/target task configurations. For the hidden state-based methods, the reported results correspond to the scenario in which the entire response is evaluated (without windowing). In the case of attention map-based methods, results are presented for a window size of 8. This reflects the default scenarios these methods were originally designed for. A more in-depth examination of the influence of window length is provided in 5.5.

5.3.1. SAPLMA

The results (see Fig. 5.2) reveal that the effectiveness of SAPLMA varies significantly across both models and tasks. Notably, LLaMA-3.1-8B emerges as the most favorable model. However, a substantial performance gap between the QA and summarization tasks is evident. The method performs considerably better on QA, achieving up to 0.9 AUC, compared to summarization, where the maximum AUC reaches only around 0.65.

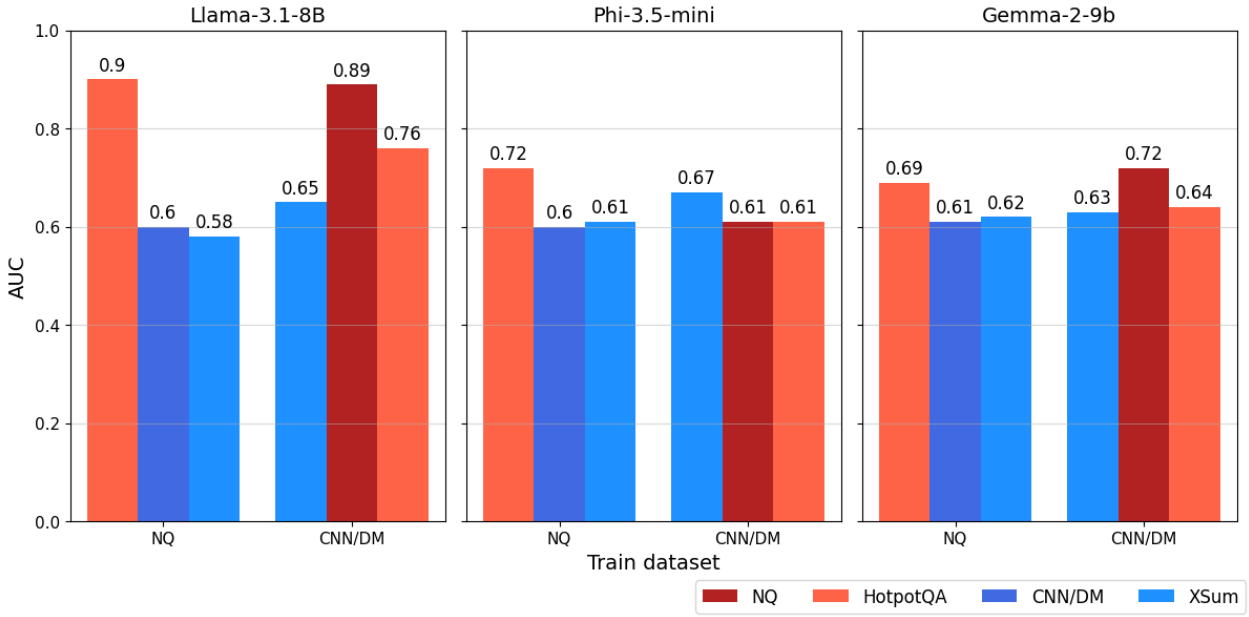

Figure 5.2: Best results obtained by SAPLMA across different LLMs and source/target configurations (no windowing).

Interestingly, this trend is consistent across both in-domain and transfer settings, suggesting that SAPLMA generalizes more effectively within the QA domain than across task types. The observed performance gap also underscores the inherent difficulty of hallucination detection in abstractive summarization, where the line between faithful generation and paraphrased hallucination is often ambiguous. These findings point to the potential need for task-specific adaptations of the method to maintain performance across diverse generation scenarios.

For Phi-3.5-mini and Gemma-2-9B, the performance gap narrows as QA results slightly decline and SUMM results improve. Nevertheless, the method still performs better on QA in 3 out of 4 cases. The AUC scores generally remain just above 0.6, while this is better than random, it may not be sufficient for high-stakes practical applications.

Overall, while SAPLMA demonstrates strong potential in QA settings—particularly with models like Llama—it appears less robust in open-ended generative tasks. This indicates that hallucination detection may not be universally transferable and could require fine-tuned calibration depending on both the target task and model architecture **(RQ3)**.

5.3.2. SEPs

The results of the SEPs method (see Fig. 5.3) are reported for the default setting of the N parameter, fixed at 10. A more in-depth analysis of how different values of N affect performance is presented in 5.6. Some similarities to SAPLMA can be noticed, such as higher performance on the QA task, particularly with the LLaMA-3.1-8B model. However, the absolute values for SEPs are lower in 13 of the 18 scenarios. In order to check whether there is a statistically significant difference between the two methods, a Wilcoxon signed-rank test was conducted. This test evaluates whether the median of the differences between paired observations is significantly different from zero. It is a non-parametric alternative to the paired t-test, used when one cannot assume the normal distribution of the differences. For a *p-value* of 0.05, the test indicated a statistically significant difference.

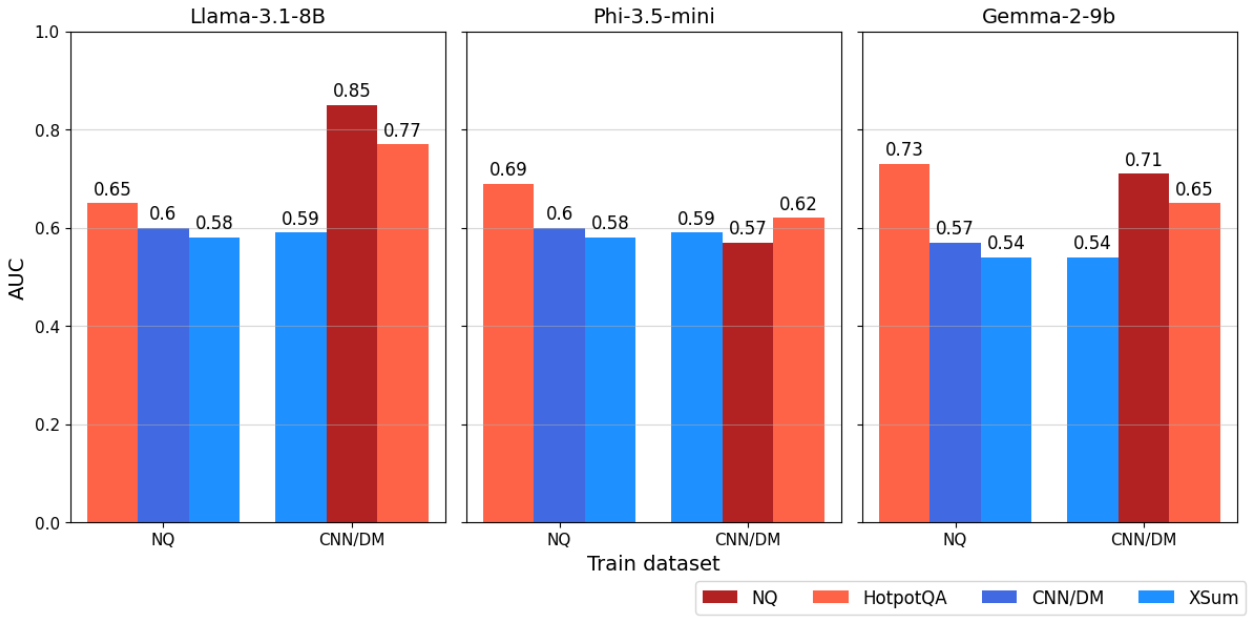

Figure 5.3: Best results obtained by SEPs across different LLMs and source/target configurations (no windowing).

Worse performance of SEPs compared to SAPLMA might stem from the fact that in the latter, the classifier is not directly trained on the hallucination detection task, but rather on the task of predicting the semantic entropy level. Investigation of this hypothesis is left for

future work. Nevertheless, it is worth noting that, despite slightly lower AUC values, SEPs offer the important advantage of not requiring labeled data. This makes SEPs particularly appealing in low-resource or deployment settings where hallucination labels are scarce or unavailable, offering a trade-off between performance and labeling cost.

5.3.3. Lookback-Lens

The most notable difference between Lookback-Lens (see Fig. 5.4) and the methods described so far lies in its comparatively better performance on the SUMM task. Unlike SAPLMA and SEPs, which show significant drops in performance when evaluated on abstractive summarization datasets, Lookback-Lens exhibits stronger generalization to this task. This holds true not only in the in-domain setting (both training and testing on SUMM), but also in the cross-task setting, when the classifier is trained on QA and evaluated on SUMM. This finding indicates that the chunk-based design of Lookback-Lens, which analyzes local token-to-token attention interactions, may help capture inconsistencies in generative content more effectively in open-ended summarization scenarios **(RQ2, RQ3)**.

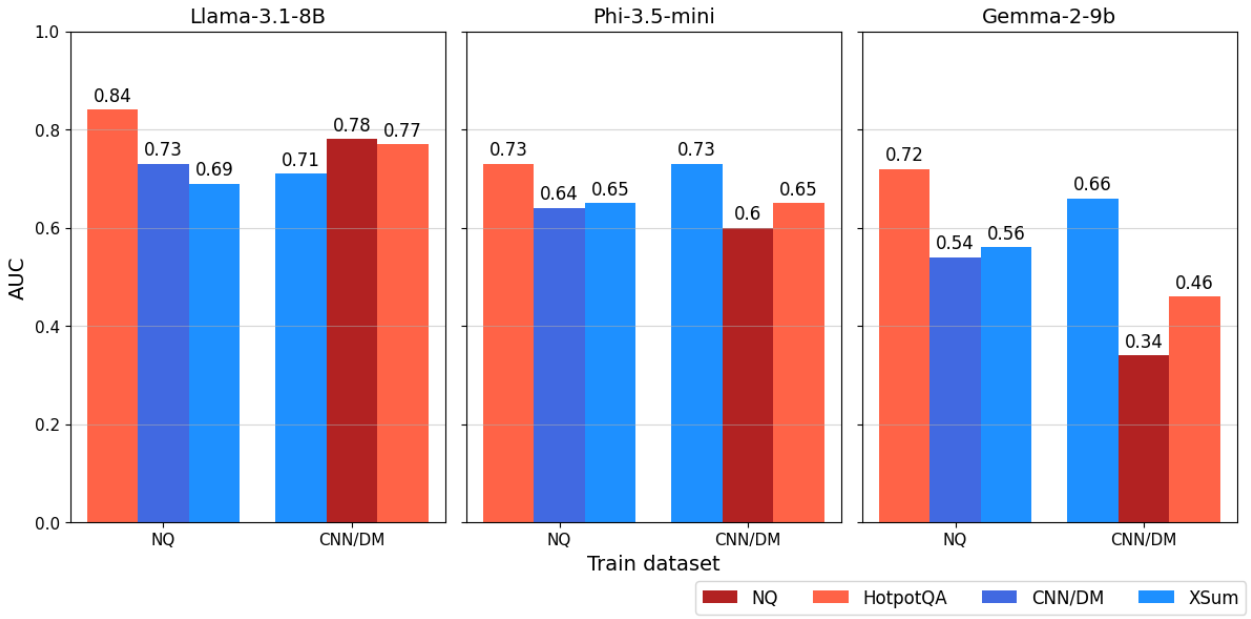

Figure 5.4: Best results obtained by Lookback-Lens across different LLMs and source/target configurations (window size equal to 8).

The performance gap between training on NQ and transferring to summarization is relatively narrow for the LLaMA-3.1-8B model, with the biggest drop of 0.07 in AUC for HotPotQA. This contrasts with the more severe degradation observed for other models such as Gemma-2-9B, where performance falls drastically (e.g., from 0.72 to just 0.46). This sharp decline suggests that Lookback-Lens is sensitive to both the model architecture and the nature of the generation task in lower-performing models.

Despite its strengths in summarization, Lookback-Lens underperforms compared to SAPLMA and SEPs when it comes to hallucination detection in the QA domain. Across

multiple models, AUC values are generally lower, especially in the SUMM to QA setting. One particularly notable case is Gemma-2-9B, where Lookback-Lens not only fails to generalize from NQ to HotPotQA, but even falls below random performance (AUC < 0.5), raising concerns about its robustness in certain transfer scenarios.

5.3.4. AggTruth

The results of the AggTruth method (see Fig. 5.5) are reported for the *JS-DIV* aggregation variant. Overall, AggTruth demonstrates a high degree of stability across both task types and across all three examined LLMs.

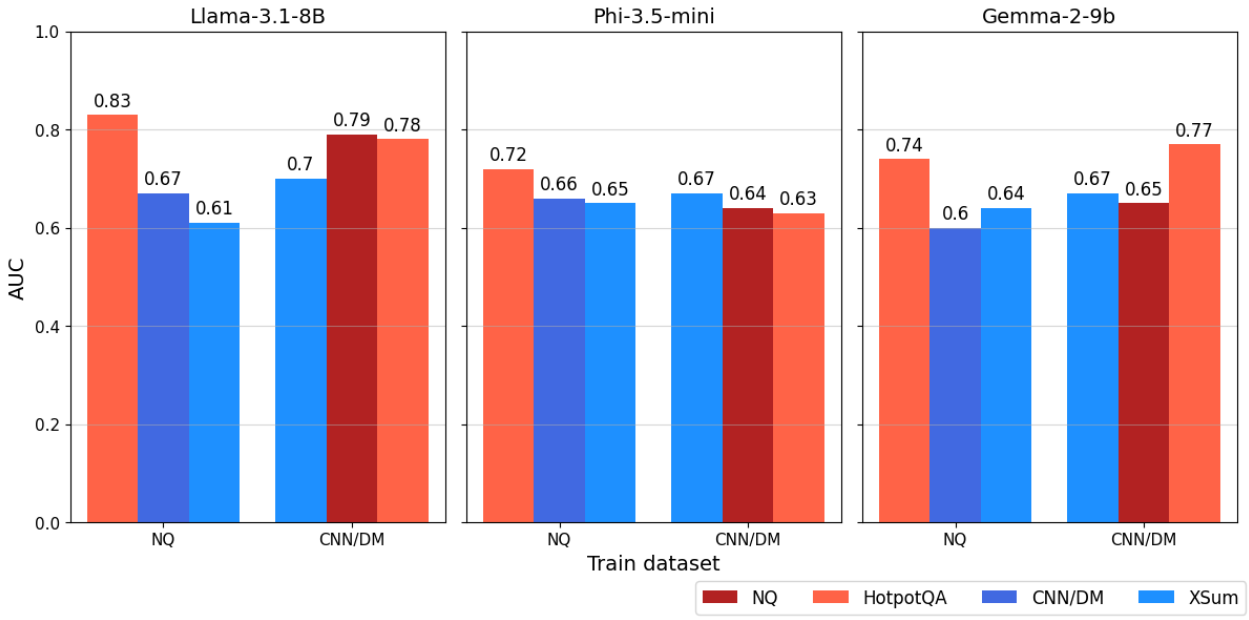

Figure 5.5: Best results obtained by AggTruth across different LLMs and source/target configurations (window size equal to 8).

AggTruth, although it generally underperforms on QA tasks compared to hidden state-based methods like SAPLMA and SEPs, particularly in configurations involving LLaMA-3.1-8B, shows notable strength on SUMM. This advantage becomes especially clear in transfer settings, where other methods often experience significant performance drops, while AggTruth maintains greater stability, demonstrating stronger generalizability.

A notable strength of AggTruth lies in its robustness across different models. Unlike Lookback-Lens, which suffers a substantial performance drop on Gemma-2-9B, especially in SUMM-to-QA transfer scenarios, AggTruth maintains competitive AUC scores even with this more challenging model. For example, it achieves an AUC of up to 0.77 on HotPotQA with Gemma-2-9B, outperforming all other methods. In comparison, Lookback-Lens performs worse than random in this setting, with an AUC below 0.5. This finding suggests that while attention patterns may carry information about answer reliability, it is crucial that the method used to extract and construct features from these patterns is either general enough to work across models or specifically tailored to the characteristics of each model.

Once again, an attention map-based method demonstrated better performance in detecting hallucinations for the SUMM task **(RQ3)**. One possible explanation for this phenomenon is that, when faced with the QA task, the model may not always rely on the provided context, but instead generate answers based on its internal knowledge. This makes analyzing the hidden states more suitable for this scenario, as [15] suggests that the MLP layers of transformers can function as key-value databases. However, when dealing with SUMM, the model is more likely to rely on the provided context, and the attention mechanism is responsible for correctly attending to the relevant portions of it. Further investigation of this hypothesis is planned for future work.

5.4. Generalizability across test datasets

One important aspect of the conducted experiments was to assess the ability of the examined hallucination detectors to generalize to different test datasets while being trained on others. This form of evaluation is crucial for understanding the robustness and practical applicability of the methods beyond their training domains.

Although performance was evaluated on three different test datasets each time, simply taking the arithmetic average of the results does not adequately reflect overall performance. This is because the datasets vary in the number of positive and negative samples they contain, as well as in their level of difficulty. These factors can skew average scores and obscure meaningful comparisons. To address this issue, a new evaluation metric called **Gap** was introduced in [30] and is defined as follows:

$$\text{Gap}_m = \frac{1}{|S|} \sum_{s \in S} \frac{\max_{m' \in M} \text{AUC}^s_{m'} - \text{AUC}^s_m}{\max_{m' \in M} \text{AUC}^s_{m'}} \times 100\%, \tag{5.1}$$

where $m \in M$ is a specific method for which Gap is calculated among all methods M considered, S is the set of all three test datasets, and AUC^s_m denotes the AUC score obtained by method m on test set s. Lower Gap values correspond to better overall generalization, indicating that the method's performance is closer to the best-performing approach on each dataset.

5.4.1. Generalizability in the windowed setting

Tab. 5.3 presents the generalization capabilities of the examined methods when applied to windows of size 8. For SAPLMA, the subscript indicates the selected layer, while for AggTruth it denotes the chosen aggregation method. Results for SEPs are omitted, as this method does not support the windowed setting. In each case, the variant of a method yielding the lowest Gap value is reported.

It can be observed that Lookback-Lens and AggTruth perform best in this setting, each achieving the lowest Gap value three times. This further highlights the importance of selecting a method that is well-suited to the specific task and configuration **(RQ2, RQ3)**. While Lookback-Lens achieves high AUC scores on same-task evaluations, its transfer performance can be less consistent, reflecting a potential overfitting to training domains compared to the more generalizable AggTruth method. Additionally, in the SUMM-to-QA configuration for

Table 5.3: Generalization capabilities of examined methods when evaluating windows of size 8.

LLM	Source	Target	Method	Same Task	Transfer (1)	Transfer (2)	Gap[%]
LLaMA-3.1-8B	QA	SUMM	SAPLMA_{16}	0.79	0.59	0.58	13.26
			Lookback-Lens	**0.84**	**0.73**	**0.69**	**0.00**
			$\text{AggTruth}_{\text{JS-DIV}}$	0.83	0.67	0.61	6.92
	SUMM	QA	SAPLMA_{12}	0.62	0.78	0.77	5.22
			Lookback-Lens	**0.71**	0.78	0.77	**0.89**
			$\text{AggTruth}_{\text{JS-DIV}}$	0.70	**0.79**	**0.78**	0.92
Phi-3.5-mini	QA	SUMM	SAPLMA_{20}	0.68	0.62	0.62	5.47
			Lookback-Lens	**0.73**	0.64	0.65	0.93
			$\text{AggTruth}_{\text{JS-DIV}}$	0.72	**0.66**	**0.65**	**0.39**
	SUMM	QA	SAPLMA_{16}	0.65	0.60	0.64	6.28
			Lookback-Lens	**0.73**	0.60	0.65	**2.47**
			$\text{AggTruth}_{\text{Entropy}}$	0.69	**0.62**	**0.65**	2.57
Gemma-2-9B	QA	SUMM	SAPLMA_{42}	**0.76**	0.57	0.55	8.23
			Lookback-Lens	0.72	0.54	0.56	10.85
			$\text{AggTruth}_{\text{JS-DIV}}$	0.75	**0.60**	**0.64**	**1.94**
	SUMM	QA	SAPLMA_{30}	0.57	0.61	0.63	13.73
			Lookback-Lens	0.66	0.34	0.46	30.80
			$\text{AggTruth}_{\text{JS-DIV}}$	**0.67**	**0.65**	**0.77**	**1.16**

Llama-3.1–8B and Phi-3.5-mini, AggTruth performs only marginally worse than Lookback-Lens, by 0.03 and 0.1 percentage points in Gap value, respectively. In contrast, in scenarios where AggTruth outperforms all other methods, Lookback-Lens lags significantly behind, with an average difference of 13.03 percentage points in Gap value.

5.4.2. Generalizability in the no windowing setting

Tab. 5.4 presents the results for the scenario without windowing. In this case, SAPLMA or SEPs achieve the best Gap value in four out of six configurations **(RQ2)**, while AggTruth performs best in the remaining cases. These results confirm that hidden state-based approaches tend to perform better in the absence of windowing, whereas attention map-based methods excel when evaluating windowed chunks. The better performance of hidden state-based methods in the non-windowed setting may be due to their ability to leverage holistic representations of entire responses, which might be diluted or fragmented when inputs are windowed.

Across both settings (windowed inputs and no windowing), AggTruth demonstrates the strongest generalization capabilities of all examined methods, as it is the only method that achieves top performance in a setting for which it was not originally designed by the authors **(RQ4)**. These findings point to the importance of further exploring adaptive or hybrid methods that can dynamically select or combine strategies depending on the input type.

Table 5.4: Generalization capabilities of examined methods when evaluating whole responses.

LLM	Source	Target	Method	Same Task	Transfer (1)	Transfer (2)	Gap[%]
LLaMA-3.1-8B	QA	SUMM	$SAPLMA_{32}$	**0.89**	**0.60**	**0.56**	**4.32**
			SEP_{10}	0.65	0.59	0.53	15.24
			Lookback-Lens	0.88	0.55	0.56	7.57
			$AggTruth_{Sum}$	0.88	0.59	0.56	5.22
	SUMM	QA	$SAPLMA_{12}$	0.65	0.87	0.74	3.45
			SEP_{26}	0.61	**0.90**	**0.78**	**2.85**
			Lookback-Lens	**0.66**	0.67	0.69	12.08
			$AggTruth_{JS\text{-}DIV}$	0.65	0.86	0.73	3.88
Phi-3.5-mini	QA	SUMM	$SAPLMA_{32}$	**0.72**	0.60	0.60	**0.93**
			SEP_{32}	0.68	0.59	0.58	4.81
			Lookback-Lens	0.68	0.57	0.61	4.33
			$AggTruth_{JS\text{-}DIV}$	0.67	**0.61**	**0.61**	2.94
	SUMM	QA	$SAPLMA_{18}$	0.66	0.59	**0.61**	3.62
			SEP_{18}	0.58	0.60	0.59	7.85
			Lookback-Lens	**0.69**	0.59	0.54	6.01
			$AggTruth_{JS\text{-}DIV}$	0.68	**0.62**	0.57	**2.73**
Gemma-2-9B	QA	SUMM	$SAPLMA_{40}$	0.74	0.55	0.56	8.07
			SEP_{42}	0.74	0.55	0.56	8.07
			Lookback-Lens	**0.77**	0.53	0.53	9.25
			$AggTruth_{Sum}$	0.75	**0.58**	**0.58**	**4.75**
	SUMM	QA	$SAPLMA_{42}$	**0.62**	0.72	0.64	**4.59**
			SEP_{30}	0.54	**0.74**	0.66	6.48
			Lookback-Lens	0.58	0.33	0.37	36.80
			$AggTruth_{CosSim}$	0.53	0.68	**0.68**	8.44

5.5. Window length influence

The goal of this experiment is to determine the extent to which dividing a model's response into chunks (with window sizes of 4, 8, 16 tokens, or no windowing) influences the performance of hallucination detection methods. This analysis enables the identification of approaches that are robust to changes in the granularity of the examined response, as well as those that require precise adjustment based on input length. The findings may offer practical guidelines for setting chunk parameters in production environments or for developing more adaptive detection frameworks. Lookback-Lens and AggTruth (JS-DIV variant) were included in the evaluation, as these methods are specifically designed to operate on chunks.

Figure 5.6 shows that window length influences the methods differently across various models and datasets. The results are reported for the scenario where the models were trained on the NQ dataset (QA task). For LLaMA-3.1-8B, performance increases when testing on the same task but decreases on the transfer task as the window length increases. For Phi-3.5-mini, the results are relatively stable, varying only slightly between window sizes of 4, 8, and 16 tokens, but performance drops when evaluating the entire response. This decline is expected, as these methods were not originally designed to assess entire responses without windowing.

Surprisingly, for Gemma-2-9B, performance actually improves between the largest window size and the case of no windowing, for both methods and across all test datasets.

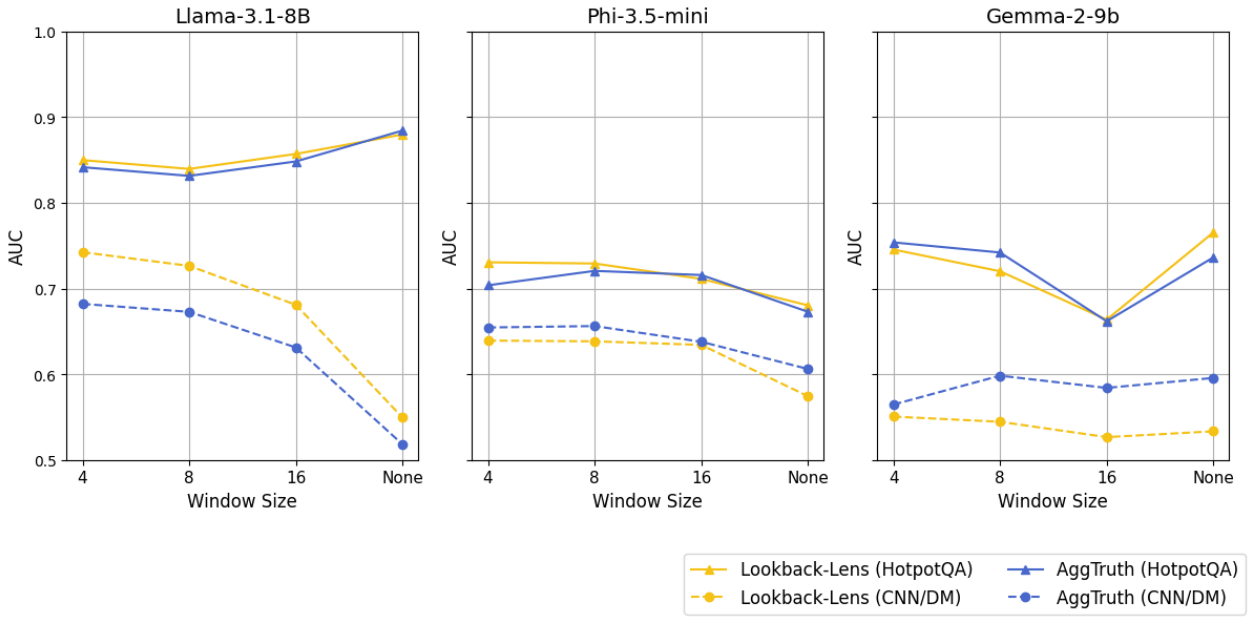

Figure 5.6: AUC of Lookback-Lens and AggTruth (JS-DIV) as a function of window length when training on NQ and evaluating on HotPotQA and CNN/DM.

A striking observation across all three models is that Lookback-Lens and AggTruth exhibit nearly identical trends — their curves almost overlap in every plot. The only consistent difference is related to the test set (in-domain vs. transfer), rather than the method itself. This suggests that while attention maps provide useful signals for hallucination detection, the specific mechanism by which they are converted into features (whether by Lookback-Lens or AggTruth) is not the dominant factor influencing the detector's performance for a given window length. Instead, performance appears more closely tied to dataset characteristics and windowing strategy.

Additionally, Fig. 5.7 presents the same experiment, but this time with models trained on CNN/DM instead of NQ and evaluated on NQ and XSum. For in-domain evaluation, the results are consistent with previous observations, with Lookback-Lens and AggTruth following nearly identical trends across all models. In the case of cross-task transfer to QA, however, the behavior of the methods diverges more significantly between models. That said, Lookback-Lens consistently underperforms compared to AggTruth, with the difference especially pronounced for Gemma-2-9B, where performance drops below random chance. This observed instability may be attributed to the fact that transfer from SUMM to QA constitutes a more difficult and less stable direction compared to the reverse, as evidenced by the analysis presented in Section 5.3.

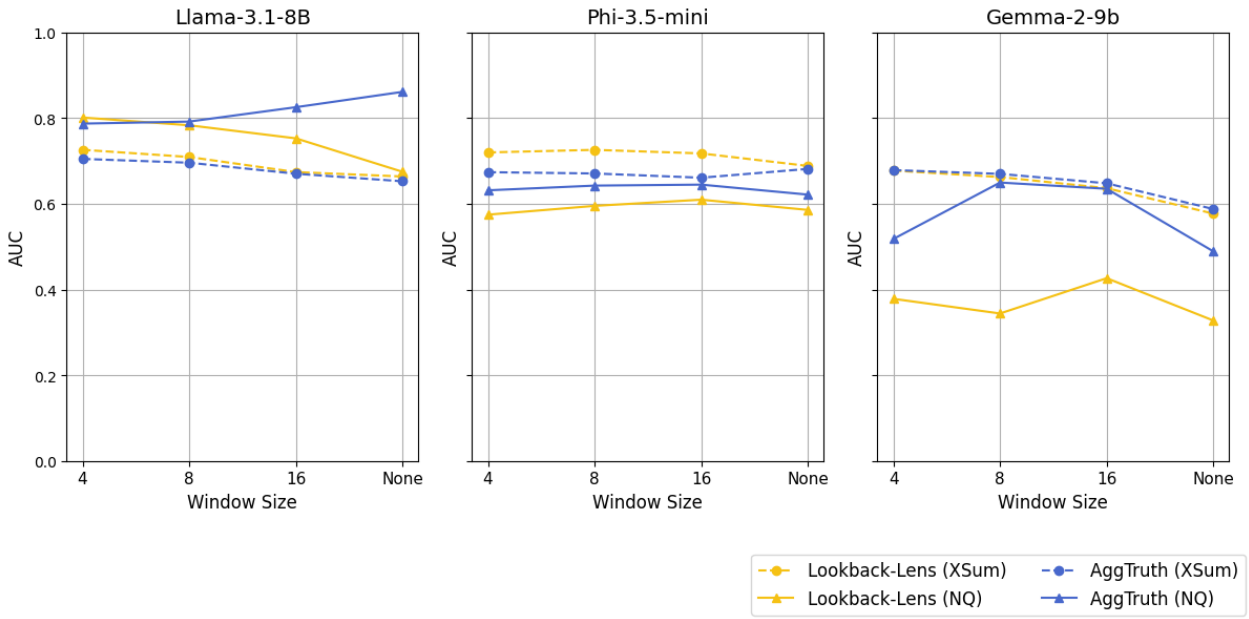

Figure 5.7: AUC of Lookback-Lens and AggTruth (JS-DIV) as a function of window length when training on CNN/DM and evaluating on NQ and XSum.

5.6. Importance of parameter N for SEPs

Recall that the parameter N defines the number of sampled responses used to estimate semantic entropy for a query when training a semantic entropy level classifier. It directly influences the computational resources required for training, which is why one would aim to keep it as low as possible. The authors proposed a value of 10.

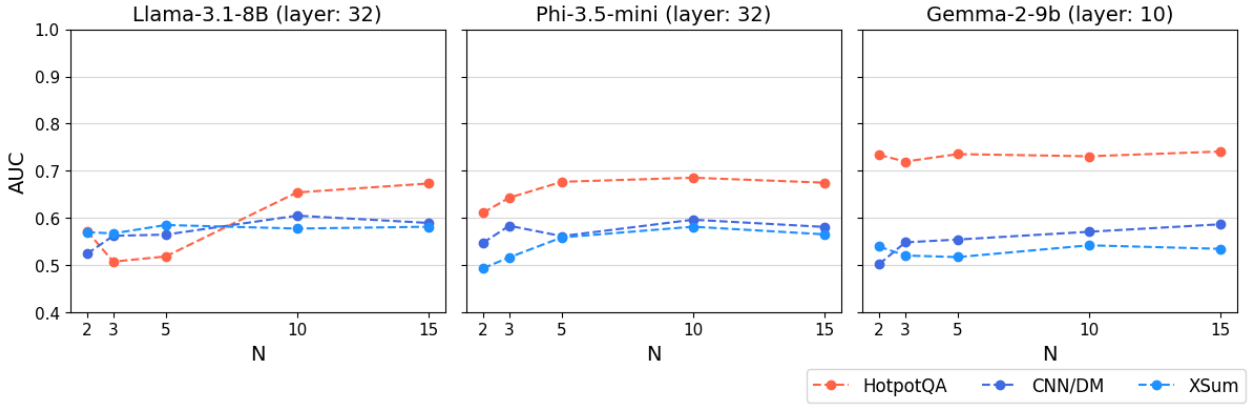

Figure 5.8: SEPs results as a function of the number of sampled responses (N) for a classifier trained on the NQ dataset.

Fig. 5.8 presents AUC scores for the scenario of training on the NQ dataset using different values of N. For each model, results are reported for the layer that performed best for that specific source/target configuration when N was set to 10. The calculations were then

repeated for other values of N, with all other parameters held constant. The results show that using a smaller value of N can lead to a degradation in performance, especially for LLaMA-3.1-8B. Conversely, increasing the number of sampled responses does not lead to any significant improvement in performance. This trade-off between computational efficiency and classification accuracy highlights the need for task-specific tuning of N. An interesting direction for future research would be to investigate adaptive sampling strategies that adjust N dynamically based on query complexity.

5.7. Optimal layer for hidden state-based methods

The performance of hidden state-based hallucination detection methods is closely tied to the choice of the transformer layer from which hidden states are extracted. This is rooted in the hierarchical nature of large language models, where different layers capture different levels of linguistic and semantic information. Consequently, identifying which layers provide the most informative signals for hallucination detection is crucial for optimizing detector performance. As part of the experiments, every second (even-numbered) layer, as well as the first layer, was analyzed for each of the considered models.

The question arises whether some layers are particularly more useful for the hallucination detection task than others. To investigate this, layers were ranked in descending order based on their performance across all three test datasets in both source/target task configurations. For SAPLMA, the experiment was conducted separately for the cases with and without windowing. The Mean Reciprocal Rank (MRR) was then used to assign each layer a score. MRR measures the average reciprocal of the rank position, according to the formula:

$$\mathrm{MRR} = \frac{1}{|S|} \sum_{i=1}^{|S|} \frac{1}{\mathrm{rank}_i} \tag{5.2}$$

where S is the set of evaluation scenarios (e.g., different task/source and test set combinations), and rank_i is the rank position of the layer in scenario i, based on AUC performance.

The calculated MRR values (see Fig. 5.9) highlight a critical distinction between scenarios with and without windowing **(RQ2)** for SAPLMA. In the case of windowed input, the final layer consistently achieves the highest performance. In contrast, for non-windowed input (for both methods), the highest AUC scores generally appear in the mid-to-late layers, but there is no single layer that stands out as definitively the best. Across all three scenarios, however, the optimal layer is always located in the second half of the model **(RQ2)**. This aligns with the recommendations provided by the authors of both methods.

Furthermore, the experiments demonstrate that choosing a suboptimal layer can markedly degrade performance, highlighting the need for careful layer selection during system design. These consistent patterns across the three examined large language models suggest a general principle applicable to similar transformer-based detectors. An interesting direction for future research would be to design mechanisms that allow for dynamic layer selection based on the characteristics of the input query.

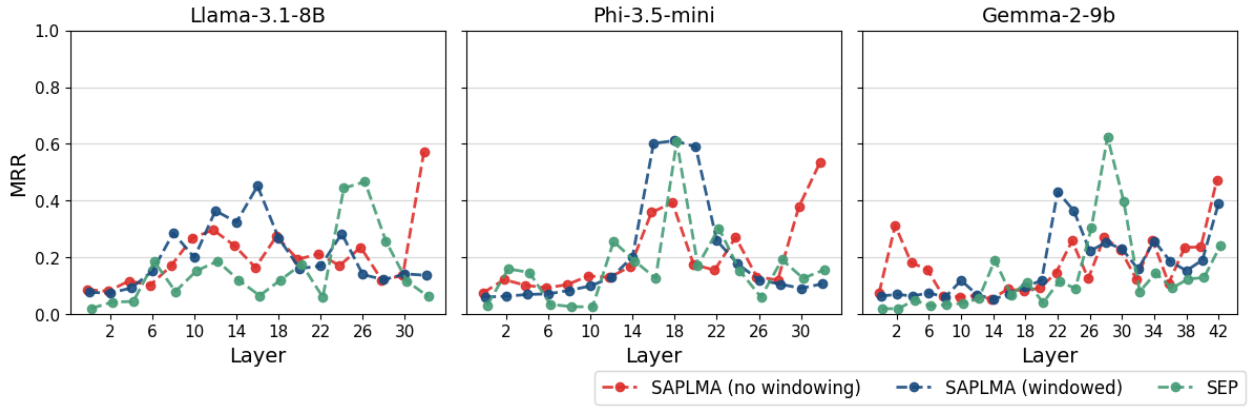

Figure 5.9: MRR of each layer for SAPLMA and SEPs methods calculated over both source/-target configurations and all test datasets.

5.8. Subsets of layers for attention map-based methods

Compared to hidden state–based methods, attention-based approaches such as Lookback-Lens and AggTruth do not treat the layer as a tunable hyperparameter. By default, they use attention maps from all layers. This raises the question of whether all layers contribute equally to encoding information relevant for hallucination detection. To assess whether useful signals are concentrated in specific parts of the network, classifiers were trained using attention heads from selected subsets of layers: four layers for LLaMA-3.1-8B and Phi-3.5-mini and three for Gemma-2-9B. The layer subsets were chosen to span early, middle, and late stages of the model to capture different levels of abstraction. The resulting AUC scores for Lookback-Lens and AggTruth (JS-DIV variant) are shown in Fig. 5.10. Models were trained on NQ and evaluated on HotPotQA and CNN/DM.

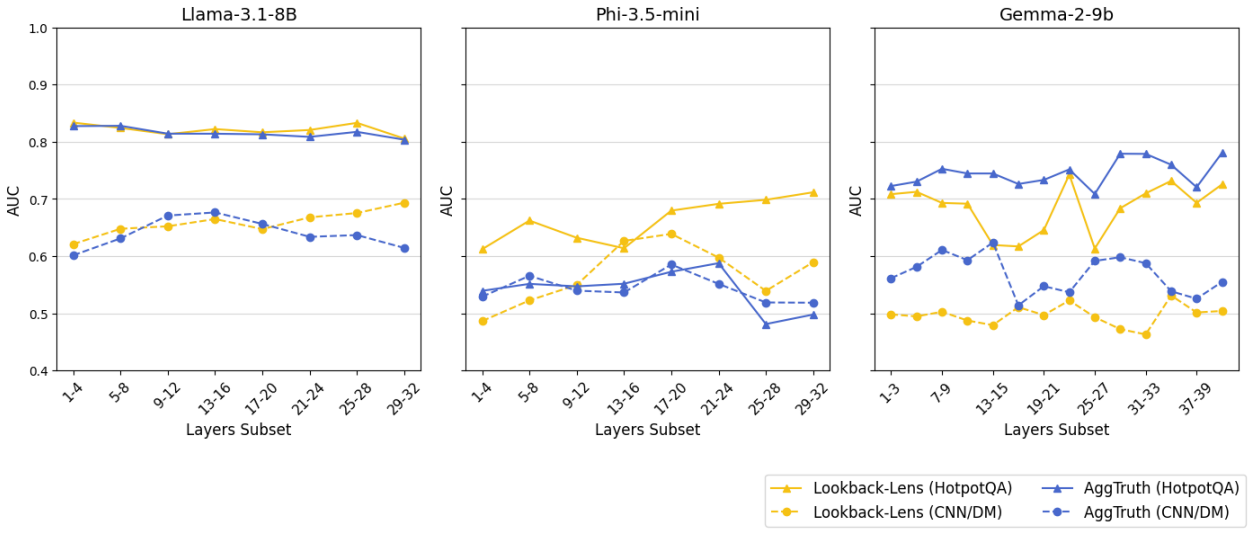

Figure 5.10: AUC of Lookback-Lens and AggTruth (JS-DIV) when training on NQ using a subset of layers and evaluating on HotPotQA and CNN/DM.

For LLaMA-3.1-8B, both Lookback-Lens and AggTruth achieve stable results regardless of the chosen subset of layers, especially on the HotPotQA task, where the AUC remains above 0.8. This suggests that, in this architecture, the information relevant to hallucination detection is evenly distributed throughout the network. However, for Phi-3.5-mini and Gemma-2-9B, significant instability is observed across different layer subsets. In the case of Phi-3.5-mini, performance drops markedly when using the top layers (25–32), suggesting that deeper layers may not contribute effectively to hallucination detection in this model. This observation aligns with trends seen in hidden state–based methods, where the final layers also performed poorly for this particular model. These findings highlight the strong model-specific nature of the hallucination detection task. This is further reinforced by the Gemma-2-9B results, where the variation is even more pronounced. For instance, when evaluating Lookback-Lens on HotPotQA, two neighboring layer subsets yield the highest (layers 22–24) and lowest (layers 25–27) AUC scores, clearly demonstrating the sensitivity of performance to layer selection in certain models.

Beyond examining how the detector's performance changes depending on the selected layer subset, another important question is whether choosing only a subset of layers improves or worsens the overall results. Table 5.5 presents a comparison of AUC scores when training the detector using all layers versus using only the best-performing subset. Interestingly, the results demonstrate that selecting an optimal subset of layers yields performance comparable to using all layers. A Wilcoxon signed-rank test (with a p-value threshold of 0.05) confirms that the observed differences are not statistically significant. While using the full set of layers remains a strong baseline, these findings suggest it is not necessary to use all layers to capture valuable information. Identifying the most informative subset of layers will be the subject of future work.

Table 5.5: Comparison of AUC results across all layers and best-performing subset of layers. Difference is calculated as subset minus full.

Model	**Eval Type**	**Method**	**AUC$_{\text{all layers}}$**	**AUC$_{\text{layers subset}}$**	**Difference**
LLaMA-3.1-8B	Same-task	Lookback	0.840	0.833	-0.007
		Agg. Truth	0.832	0.828	-0.004
	Transfer	Lookback	0.727	0.693	-0.034
		Agg. Truth	0.673	0.676	+0.003
Phi-3.5-mini	Same-task	Lookback	0.729	0.712	-0.017
		Agg. Truth	0.721	0.588	-0.133
	Transfer	Lookback	0.639	0.639	0.000
		Agg. Truth	0.656	0.585	-0.071
Gemma-2-9B	Same-task	Lookback	0.720	0.742	+0.022
		Agg. Truth	0.742	0.781	+0.039
	Transfer	Lookback	0.545	0.531	-0.014
		Agg. Truth	0.599	0.624	+0.025

5.9. Comparison of AggTruth aggregation methods

The aim of this experiment is to determine whether there exists a single optimal aggregation method within AggTruth, as the authors proposed four different variants. Figure 5.11 presents the ranking of these methods based on the Mean Reciprocal Rank (MRR) metric (see Eq. 5.2). For each possible experimental scenario (the Cartesian product of models, test datasets, and window sizes), the four examined methods were ranked, and the MRR was calculated over all such rankings. The results demonstrate that the JS-Div method significantly outperforms its competitors, while the differences among the remaining methods are more subtle.

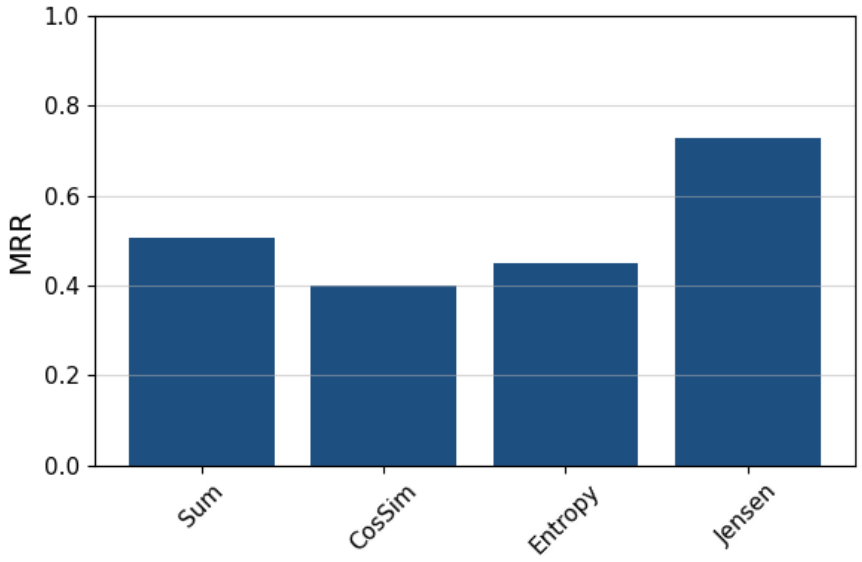

Figure 5.11: MRR of AggTruth aggregation methods calculated on three examined models, both source/target configurations, and all window sizes.

Tab. 5.6 presents a more in-depth comparison. It reveals that the JS-DIV aggregation method is the most effective in terms of generalization across tasks, achieving the highest results in most cases and minimizing the gap between performance on the same task and transfer performance. Compared to other methods like CosSim or Entropy, JS-DIV demonstrates greater stability and better transfer results, regardless of model architecture or transfer direction. In two cases, it is able to achieve the perfect Gap value equal to 0.00%. Importantly, JS-DIV's effectiveness appears largely independent of model architecture. It demonstrates stable generalization across lightweight models like Phi-3.5-mini as well as larger ones like LLaMA-3.1-8B and Gemma-2-9B, underscoring its broad applicability. This architecture-agnostic behavior suggests that JS-DIV is a reliable and scalable choice for multi-task and transfer learning settings. Therefore, JS-DIV should be the preferred aggregation method in practical applications where robustness to task changes is crucial.

In contrast, methods such as CosSim and Entropy exhibit greater variability and are more susceptible to domain shift. Notably, CosSim yields a Gap value of 20.67% in the SUMM to QA cross-task scenario for Gemma-2-9B, while Entropy performs even worse, exhibiting a Gap of 25.40%, which represents the largest degradation observed across the entire evaluation.

Table 5.6: Generalization performance of AggTruth aggregation methods across tasks.

Model	Source	Target	Agg. Method	Same Task	Transfer (1)	Transfer (2)	Gap [%]
LLaMA-3.1-8B	QA	SUMM	Sum	0.81	0.65	0.63	2.31
			CosSim	0.81	0.62	**0.63**	3.45
			Entropy	0.81	0.61	0.60	5.61
			JS-DIV	**0.83**	**0.67**	0.61	**1.31**
	SUMM	QA	Sum	0.69	0.72	0.76	4.38
			CosSim	0.65	0.70	0.66	10.79
			Entropy	0.69	0.70	0.75	5.49
			JS-DIV	**0.70**	**0.79**	**0.78**	**0.00**
Phi-3.5-mini	QA	SUMM	Sum	0.71	0.60	0.56	7.92
			CosSim	0.66	0.57	0.54	12.91
			Entropy	0.71	0.61	0.58	6.17
			JS-DIV	**0.72**	**0.66**	**0.65**	**0.00**
	SUMM	QA	Sum	0.68	0.61	0.64	2.60
			CosSim	0.63	0.62	0.63	5.15
			Entropy	**0.69**	0.62	**0.65**	**0.93**
			JS-DIV	0.67	**0.64**	0.63	1.91
Gemma-2-9B	QA	SUMM	Sum	0.75	0.60	**0.64**	**1.94**
			CosSim	**0.79**	**0.61**	0.59	2.92
			Entropy	0.74	0.60	0.63	3.67
			JS-DIV	0.74	0.60	0.64	3.00
	SUMM	QA	Sum	0.69	0.58	0.70	6.98
			CosSim	0.60	0.53	0.54	20.67
			Entropy	**0.69**	0.45	0.42	25.40
			JS-DIV	0.67	**0.65**	**0.77**	**1.16**

This indicates that although these methods may be competitive under controlled laboratory conditions, they cannot be reliably deployed in practice due to the unpredictability of real-world data. Additionally, while the Sum method demonstrates moderate stability, it fails to match the transfer efficiency of JS-DIV. Although it occasionally attains high scores (e.g., Gemma-2-9B, QA $\rightarrow$ SUMM, with a Gap of only 1.94%), it lacks the cross-task consistency exhibited by JS-DIV.

These results confirm that the choice of aggregation method can have a substantial impact on transferability. Therefore, a careful selection should be considered a critical component when implementing the method in a real-world setting.

6. Conclusions and Future Work

The experiments conducted in this study reveal that the performance of hallucination detection methods varies significantly depending on the examined LLM, the source/target configuration, as well as whether the entire response or windows are evaluated. There is no single answer to the question: *Which of the two, hidden states or attention maps, are more suitable as features for detecting hallucinated responses?* as the performance of different methods varies significantly depending on numerous factors.

Both hidden state-based and attention map-based methods exhibit notable variability depending on the specific layers used for analysis **(RQ1)**. Layer selection is a more critical factor for hidden state-based approaches, where it acts as a tunable hyperparameter. For hidden state-based methods (i.e., SAPLMA and SEPs), the optimal layer for hallucination detection was consistently found in the second half of the transformer model. This suggests that earlier layers may encode more syntactic features, while later layers better capture semantic information relevant to hallucination detection. Importantly, selecting suboptimal layers can lead to significant performance degradation, underscoring the importance of informed layer selection. Future work should explore automated strategies for identifying the most informative layers for hallucination detection, potentially improving robustness and reducing the need for extensive manual tuning.

In contrast, attention map-based methods (i.e., Lookback-Lens and AggTruth) do not treat individual layers as hyperparameters but instead aggregate information across all layers by default. However, to investigate whether certain layers contribute more significantly than others, experiments were conducted in which classifiers were trained on selected subsets of layers. The findings indicate that the useful signals for hallucination detection are not uniformly distributed across all layers. For certain models, particularly Phi-3.5-mini and Gemma-2-9B, performance varied significantly depending on the selected subset of layers, suggesting that deeper layers may not always contribute effectively to hallucination detection. Interestingly, selecting only an optimal subset of layers yielded performance comparable to using all layers. This implies that it may not be necessary to use all layers to extract valuable information, opening up future research directions focused on efficient layer selection and feature engineering.

One of the most distinguishing characteristics between hidden state-based and attention map-based methods is their behavior in relation to windowing **(RQ2)**. Hidden state-based methods outperform attention-based methods when evaluating full responses, while attention-based methods perform better when assessing windowed responses. This aligns with the original design goals of these methods and the assumptions made in their respective papers. Hidden state-based methods rely on a final token representation that aggregates reliability information across the entire response. When responses are divided into windows, the final token within each window no longer retains the same holistic properties, leading to performance degradation. In contrast, attention map-based methods construct features based on all tokens within a window, allowing them to better capture inconsistencies.

Another key finding is the task-specific effectiveness of different method types **(RQ3)**. Hidden state-based methods generally perform better on question answering (QA) tasks, while attention map-based methods excel in abstractive summarization (SUMM). A hypothesis presented in this study attributes this difference to how models handle context during generation: in QA tasks, models often rely on internal knowledge rather than provided context, making hidden states more informative. In contrast, SUMM tasks require the model to attend closely to input context, making attention mechanisms more indicative of hallucination likelihood. Further investigation into this hypothesis is planned for future work.

Across all models and configurations, methods tend to perform better when detecting hallucinations in QA tasks compared to SUMM. The gap is particularly pronounced for hidden state-based methods. While attention map-based methods show a narrower performance gap, they still exhibit reduced effectiveness in SUMM. This can be attributed to the nature of abstractive summarization, where generated content may contain both faithful and hallucinated segments, making it challenging to evaluate the overall reliability of the output. This poses a particular challenge for hidden state-based methods, which are designed to assess complete responses. In such cases, attention map-based methods that evaluate response fragments may offer greater precision in identifying hallucinated content.

The examined LLM has a substantial impact on the performance of hallucination detection methods. All methods achieved the best results with LLaMA-3.1-8B, and for this model, the selection of layer subsets in attention-based methods was most stable. Phi-3.5-mini delivered weaker overall results but exhibited high stability across different window sizes and offered a larger margin of error when selecting the optimal layer for hidden state-based methods. Gemma-2-9B was the most unpredictable model. Hidden state-based methods performed only marginally better than random classification in SUMM, while Lookback-Lens performed even worse than random in QA. Only AggTruth demonstrated stable performance on this model. Additionally, results for Gemma-2-9B showed significant variation depending on the selected layer subset for attention map-based methods, likely due to its unique architecture (42 layers with 16 heads each), which differs from LLaMA-3.1-8B and Phi-3.5-mini models (both 32 layers with 32 heads each).

A central part of this work is the introduction and evaluation of the AggTruth method, which demonstrates superior generalization capabilities compared to Lookback-Lens, the main existing attention-based method. This is especially important in real-world scenarios where the distribution of inputs is unknown or variable. Among the four aggregation variants tested in AggTruth, JS-DIV emerged as the most effective, achieving the highest AUC scores across most test datasets and minimizing the performance gap between in-domain and cross-task settings. Notably, JS-DIV exhibited consistent performance regardless of model architecture. These findings suggest that JS-DIV is a reliable and scalable option for multi-task environments. In contrast, other aggregation methods such as CosSim and Entropy were more susceptible to domain shifts and showed higher variability. The Sum method, while moderately stable, failed to match the cross-task consistency of JS-DIV.

One of the key advantages of AggTruth lies in its design, which enables on-the-fly evaluation of generated responses in a windowed fashion, analyzing content chunk by chunk as it is being generated. This capability opens up new possibilities for real-time hallucination-aware decoding strategies, where the generation process can be dynamically guided by the hallucination detector. Instead of merely evaluating a fully generated response post hoc, future work could

explore integrating AggTruth directly into the decoding mechanism to influence token selection based on hallucination likelihood. Such an approach would represent a significant step forward in improving the reliability of LLMs.

In summary, there is no one-size-fits-all hallucination detection method that performs optimally across all scenarios **(RQ4)**. The choice of method should be tailored to the specific model, data distribution, and whether the goal is to evaluate full responses or windowed chunks. That said, AggTruth (specifically the JS-DIV variant) stands out for its strong generalization capabilities and is recommended when the deployment environment is variable.

Bibliography

[1] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[2] A. Amayuelas, K. Wong, L. Pan, W. Chen, and W. Y. Wang. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6416–6432, 2024.

[3] A. Azaria and T. Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, 2023.

[4] J. Binkowski, D. Janiak, A. Sawczyn, B. Gabrys, and T. Kajdanowicz. Hallucination detection in llms using spectral features of attention maps. *arXiv preprint arXiv:2502.17598*, 2025.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] M. Cao, Y. Dong, and J. Cheung. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022.

[7] C. Chen, K. Liu, Z. Chen, Y. Gu, Y. Wu, M. Tao, Z. Fu, and J. Ye. Inside: Llms' internal states retain the power of hallucination detection. In *ICLR*, 2024.

[8] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[9] Y.-S. Chuang, L. Qiu, C.-Y. Hsieh, R. Krishna, Y. Kim, and J. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1419–1436, 2024.

[10] Y.-S. 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 *The Twelfth International Conference on Learning Representations*, 2023.

[11] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, 2022.

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

[13] K. Filippova. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 864–870, 2020.

[14] M. Geva, J. Bastings, K. Filippova, and A. Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, 2023.

[15] M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[16] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[18] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[19] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

[21] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

[22] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[23] T. Kocmi and C. Federmann. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, 2023.

[24] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz, et al. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, 2023.

[25] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[26] J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*, 2024.

[27] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[28] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[29] P. Manakul, A. Liusie, and M. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, 2023.

[30] 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. *ICCS*, 2025.

[31] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, 2020.

[32] T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, et al. Gemma: Open models based on gemini research and technology. *CoRR*, 2024.

[33] S. Narayan, S. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics, 2018.

[34] OpenAI. ChatGPT. https://openai.com/index/chatgpt, 2022. [Online; accessed 5-May-2025].

[35] H. Orgad, M. Toker, Z. Gekhman, R. Reichart, I. Szpektor, H. Kotek, and Y. Belinkov. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707*, 2024.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[37] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. T. I. Tonmoy, A. Chadha, A. Sheth, and A. Das. The troubling emergence of hallucination in large language models-an extensive definition, quantification, and prescriptive remediations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2541–2573, 2023.

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

[39] A. Slobodkin, O. Goldman, A. Caciularu, I. Dagan, and S. Ravfogel. The curious case of hallucinatory (un) answerability: Finding truths in the hidden states of over-confident large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3607–3625, 2023.

[40] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[41] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] B. Weiser. Lawyer who used ChatGPT faces penalty for made up citations. https://www.nytimes.com/2023/06/08/nyregion/lawyer-chatgpt-sanctions.html, 2023. [New York Times; Online; accessed 5-May-2025].

[44] Z. Xu, S. Jain, and M. Kankanhalli. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*, 2024.

[45] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[46] O. Yoran, T. Wolfson, O. Ram, and J. Berant. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations*, 2024.

[47] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer, 2024.

[48] M. Yuksekgonul, V. Chandrasekaran, E. Jones, S. Gunasekar, R. Naik, H. Palangi, E. Kamar, and B. Nushi. Attention satisfies: A constraint-satisfaction lens on factual

errors of language models. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

[49] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith. How language model hallucinations can snowball. In *International Conference on Machine Learning*, pages 59670–59684. PMLR, 2024.

List of Figures

List of Tables